



# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

## ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

## MOŽNOSTI VOLNĚ DOSTUPNÉ UMĚLÉ INTELIGENCE

OPEN SOURCE ARTIFICIAL INTELLIGENCE OPTIONS

### BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

#### AUTOR PRÁCE

AUTHOR

LUBOMÍR OSTRÝ

#### VEDOUCÍ PRÁCE

SUPERVISOR

Ing. MARTIN APPEL

BRNO 2018



## Zadání bakalářské práce

Ústav: Ústav mechaniky těles, mechatroniky a biomechaniky  
Student: **Lubomír Ostrý**  
Studijní program: Aplikované vědy v inženýrství  
Studijní obor: Mechatronika  
Vedoucí práce: **Ing. Martin Appel**  
Akademický rok: 2017/18

Ředitel ústavu Vám v souladu se zákonem č.1111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

### Možnosti volně dostupné umělé inteligence

#### Stručná charakteristika problematiky úkolu:

Cílem práce je průzkum dostupné umělé inteligence, která má svůj program volně přístupný nebo ji lze používat pomocí API. Výstupem práce bude seznam dostupné umělé inteligence nebo neuronových sítí a návod jejího využití ve svém programu. Práce se také bude zabývat dostupnými daty pro učení neuronových sítí.

#### Cíle bakalářské práce:

- 1) Provedte rešerši v oblasti volně dostupné umělé inteligence.
- 2) Provedte rešerši v oblasti dostupných dat pro učení neuronových sítí.
- 3) Vytvořte seznam ukázek vybraných programů.
- 4) Aplikujte některý z těchto nástrojů do své aplikace.

#### Seznam doporučené literatury:

SILLA, Carlos N., Marcelo PAGLIONEY a Iuri G. P. MARDEGANY. JOthelloT: A java-based open source Othello framework for artificial intelligence undergraduate classes. In: 2016 IEEE Frontiers in Education Conference (FIE) [online]. IEEE, 2016, s. 1-7 [cit. 2017-10-25]. DOI: 10.1109/FIE.2016.7757577. ISBN 978-1-5090-1790-4. Dostupné z: <http://ieeexplore.ieee.org/document/7757577/>

Datamation [online]. [cit. 2017-10-25]. Dostupné z: <https://www.datamation.com/open-source/slideshows/15-top-open-source-artificial-intelligence-tools.html>

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2017/18

V Brně, dne

L. S.

---

prof. Ing. Jindřich Petruška, CSc.  
ředitel ústavu

---

doc. Ing. Jaroslav Katolický, Ph.D.  
děkan fakulty

## **Abstrakt**

Tato práce je zaměřena na volně dostupné nástroje a zdroje z oblasti umělé inteligence, konkrétně z odvětví machine learning. Jejím úkolem je prozkoumat současný stav, možnosti a omezení práce s open source programy umělé inteligence. První část práce popisuje vybrané základní pojmy týkající se machine learningu zejména neuronové sítě, jejich učení a využití. Další sekce se zaměřuje na nástroje pro machine learning, jsou zde popsány charakteristiky jednotlivých programů, kompatibilita a jejich využití. Zdroje volně přístupných dat pro učení neuronových sítí jsou tématem další z kapitol. Na ukázkou byl vytvořen program používající některé z popsaných nástrojů demonstrující jejich možnosti a využití.

## **Summary**

This thesis focuses on open source tools and resources in the field of artificial intelligence, particularly in machine learning. The aim is to analyze current state, possibilities and limitations of work with a set of open source artificial intelligence programs. The first part describes and explains basic terms regarding machine learning, mainly neural networks, their training and use. Following section describes a set of machine learning tools, specifically their main characteristics, compatibility and use. Available sources of open source datasets for neural network training is a topic of another chapter. Lastly an application was created using a selection of described tools displaying their possibilities and use.

## **Klíčová slova**

Umělá inteligence, machine learning, neuronové sítě, open source software

## **Keywords**

Artificial intelligence, machine learning, neural networks, open source software

OSTRÝ, L. *Možnosti volně dostupné umělé inteligence*. Brno: Vysoké učení technické v Brně, Fakulta strojího inženýrství, 2018. 33 s. Vedoucí Ing. Martin Appel.

Prohlašuji, že jsem bakalářskou práci *Možnosti volně dostupné umělé inteligence* vypracoval samostatně pod vedením Ing. Martina Appela, s použitím materiálů uvedených v seznamu literatury.

Lubomír Ostrý



Děkuji mému vedoucímu Ing. Martinu Appelovi za jeho čas a odborné vedení mé bakalářské práce.

Lubomír Ostrý

# Obsah

<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Základní pojmy</b>	<b>3</b>
2.1	Neuronové sítě	3
2.2	Základní typy neuronových sítí	3
2.3	Učení neuronových sítí	5
2.4	Hlavní typy úloh	5
<b>3</b>	<b>Volně dostupné nástroje pro umělou inteligenci</b>	<b>6</b>
3.1	Caffe	6
3.2	Deeplearning4j	7
3.3	H2O	7
3.4	Microsoft Cognitive Toolkit (CNTK)	8
3.5	Theano	9
3.6	Mahout	9
3.7	MLib	10
3.8	DLib	10
3.9	NuPIC	11
3.10	TensorFlow	12
3.11	Torch	12
<b>4</b>	<b>Volně dostupná data pro učení neuronových sítí</b>	<b>14</b>
4.1	Kaggle	14
4.2	UCI Machine Learning Repository	15
4.3	Další zdroje volně dostupných dat pro machine learning	15
<b>5</b>	<b>Ukázky vybraných nástrojů</b>	<b>16</b>
5.1	Caffe	16
5.2	Tensorflow	16
5.3	Dlib	17
5.4	Theano	18
<b>6</b>	<b>Aplikace</b>	<b>20</b>
6.1	Volba prostředí a nástrojů	20
6.2	Funkcionalita	20
6.3	Struktura aplikace	21
6.4	Předběžné zpracování obrazu	21
6.5	Model	22
6.6	Učení modelu, dataset	23
6.7	Vytváření seznamu známých obličejů	24
6.8	Načítání externích dat	24
6.9	Vyhodnocování podobnosti obličejů	24
6.10	Hodnocení aplikace	25
<b>7</b>	<b>Závěr</b>	<b>26</b>

# 1. Úvod

Oblast umělé inteligence nabízí mnoho open source nástrojů a zdrojů pro tvoření AI aplikací. Jedná se o rychle rozvíjející se odvětví a rychle roste i počet aplikací pro běžné použití využívající nějakou formu umělé inteligence. V současnosti je uživateli umožněna tvorba takových aplikací s použitím pouze open source softwaru.[1]

Umělá inteligence je široký pojem, tato práce se zabývá hlavně odvětvím zvaným machine learning. Machine learning se odlišuje od ostatních metod umělé inteligence učením. Výsledný model umělé inteligence tedy není definovaný explicitně, ale pouze implicitně skrze dodaná data pro učení. Kvalita výsledného modelu z toho důvodu silně závisí také na kvalitě dat pro učení, volně dostupné zdroje kvalitních dat jsou proto velmi důležité.[52]

Některé nástroje popsáné v této práci jsou určeny pro výzkumnou činnost, některé byly vyvinuty za komerčními účely, ale všechny v jádře stojí na stejných teoretických základech. Toto poukazuje na univerzálnost použití umělé inteligence.

Většina zde zmíněných programů funguje na podobném principu bez ohledu na použití. Umožňují uživateli vytvořit model umělé inteligence na bázi neuronových sítí a vycvičit jej pomocí dodaných dat. Rozdíly mezi jednotlivými programy pak tkví zejména v míře, ve které může uživatel do celého procesu zasahovat. Nástroje umožňující uživateli vysokou přizpůsobitelnost jsou ze své podstaty složitější a méně uživatelsky přívětivé.

Cíl této práce je přiblížení současného stavu nástrojů pro vytváření machine learning aplikací. Vzhledem k jejich velkému množství a rozmanitosti má uživatel mnoho možností a volba nástrojů závisí na požadavcích uživatele a aplikace. Tato práce popisuje některé z těchto nástrojů a zabývá se jejich rozdíly, charakteristickými vlastnostmi a u některých je jejich použití prakticky demonstrováno vytvořením AI aplikace.

Kromě samotných programů jsou pro tvorbu aplikací umělé inteligence důležité i další zdroje, jako například veřejně dostupné neuronové sítě, u kterých je empiricky ověřena vhodnost pro danou aplikaci, nebo také již hotové modely připravené k implementaci. Všechny tyto volně dostupné nástroje dohromady vytváří prostředí, které umožňuje rychlý vývoj machine learning aplikací.

## 2. Základní pojmy

Tato práce se zabývá open source programy pro tvoření aplikací umělé inteligence, které využívají předně metodu zvanou machine learning, neboli strojové učení. Machine learning je oblast umělé inteligence zaměřující se na napodobování, pochopení, nebo jak název napovídá, učení z poskytnutých dat. V nejširším smyslu se machine learning algoritmy snaží nalézt trendy a procesy v datech ze kterých se učí a napodobit je. Na rozdíl od jiných metod tvoření aplikací umělé inteligence závisí machine learning AI modely silně na datech učení a nejsou explicitně navrženy. Machine learning lze využít k napodobování procesů mozku, jako například vidění a rozpoznání řeči, ale také je možné pomocí této metody takovéto procesy obohatit. Další častá aplikace machine learning metod jsou úlohy předpovědi.[52]

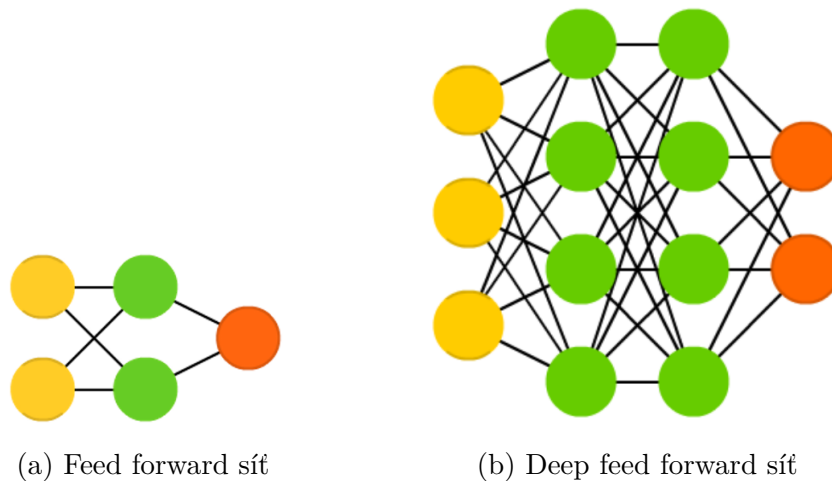
### 2.1. Neuronové sítě

Nejběžnější způsob realizace machine learning aplikací je pomocí umělých neuronových sítí. V základu je umělá neuronová síť matematický výpočetní model volně inspirovaný biologickými neuronovými sítěmi. Základní stavební jednotka neuronových sítí je neuron, spojení mezi jednotlivými neurony potom představuje matematickou operaci aplikovanou na data předávaná mezi neurony. Neurony jsou seskupovány do vrstev a dále kombinací vrstev vzniká neuronová síť. Existuje řada typů a architektur neuronových sítí s různými vlastnostmi pro různé aplikace. Kombinacemi neuronových sítí různých druhů dále dosahujeme nových vlastností.[55]

### 2.2. Základní typy neuronových sítí

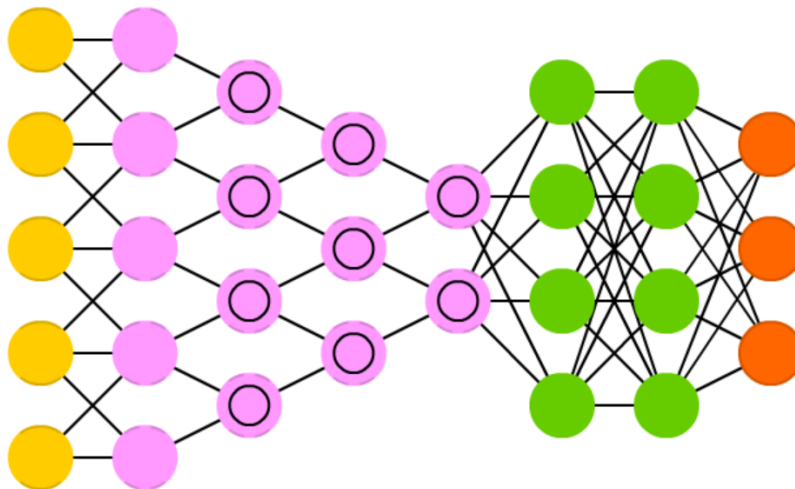
Základní typ umělé neuronové sítě je tzv. feed forward síť (obrázek 2.1a). Jedná se o síť se širokou možností využití vhodnou pro mnoho typů aplikací. Feed forward síť se skládá ze vstupní vrstvy, výstupní vrstvy a mezi nimi z libovolného počtu tzv. plně propojených vrstev. Plně propojená vrstva je vrstva neuronů, ve které jsou jednotlivé neurony propojené s každým neuronem v předchozí i nadcházející vrstvě. S narůstajícím počtem plně propojených vrstev a neuronů v nich, roste i schopnost zvládat složitější úkoly. Feed forward sítím s velkým počtem vrstev se někdy také říká deep feed forward síť (obrázek 2.1b). Nevýhodou tohoto typu sítě je vysoká výpočetní náročnost, rostoucí exponenciálně s rozšiřováním sítě, z tohoto důvodu jsou feed forward sítě kombinovány s jinými typy sítí pro zvýšení efektivity.[54]

## 2.2. ZÁKLADNÍ TYPY NEURONOVÝCH SÍTÍ



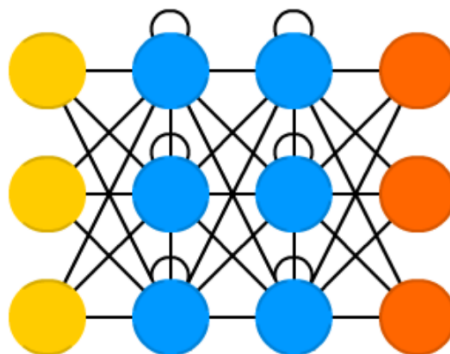
Obrázek 2.1: Feed forward neuronové síť [54]

Dalším často používaným typem umělých neuronových sítí jsou konvoluční neuronové síť (obrázek 2.2). Konvoluční neuronové síť jsou používány především pro zpracování obrazu, lze je ale používat také pro jiné typy vstupů, jako například audio data. V případě zpracování obrazu detekuje tento typ sítí metodou konvoluce naučené rysy, kdy jemné rysy jsou detekovány v úvodních vrstvách a při postupu do dalších vrstev jsou detekovány postupně hrubší rysy. Výstup z konvolučních vrstev sítě je klasifikován pomocí feed forward síť.[54]



Obrázek 2.2: Konvoluční neuronová síť [54]

Rekurentní neuronové síť (obrázek 2.3) se liší od jiných typů uvažováním zpětné vazby mezi jednotlivými průchody danou sítí. Na rozdíl od jiných typů sítí u rekurentních sítí tedy nezáleží pouze na aktuálním vstupu ale i na historii předchozích vstupů, toto v praxi znamená, že jednotlivé neurony mají paměť. Rekurentní neuronové síť jsou proto používány pro data s časovou závislostí, kde podle historie dat dokáže síť předpovídat následující data.[54]



Obrázek 2.3: Rekurentní neuronová síť [54]

### 2.3. Učení neuronových sítí

Učení neuronových sítí je možné rozdělit do dvou hlavních kategorií, učení s učitelem a učení bez učitele.[51]

Učení s učitelem je proces učení, při kterém poskytujeme neuronové síti vstup a příslušný požadovaný výstup. Pomocí tohoto postupu neuronová síť „hledá“ vztah mezi vstupními a výstupními hodnotami a vytváří se tak model umělé inteligence, který pomocí tohoto vztahu předpovídá výstupní data. Metoda učení s učitelem je více rozšířená a efektivnější při vytváření modelů umělé inteligence.[51]

Metoda učení bez učitele je „náročnější“ v porovnání s metodou učení s učitelem, jelikož při učení bez učitele neposkytujeme neuronové síti požadovaný výstup na daný vstup. Tato metoda učení je například vhodná k analýze předem neznámého vztahu mezi poskytovanými daty. V závislosti na konkrétní aplikaci je používáno učení s nebo bez učitele.[51]

### 2.4. Hlavní typy úloh

Regrese je typ úlohy, při které model umělé inteligence přiřazuje vstupu číselnou hodnotu z určitého intervalu. Posuzujeme zde tedy kvantitativní vlastnost a popisujeme tuto vlastnost číselnou hodnotou. Příklad úlohy regrese může být například určování věku člověka z fotografie. Typicky se používá učení s učitelem pro úlohy regrese.[51]

Klasifikace je typ úlohy, při které model umělé inteligence přiřazuje vstupní data do předem definovaných tříd. Jedná se tedy o úlohu, kde posuzujeme kvalitativní vlastnosti zkoumaných dat. Jako klasifikační typ úlohy bychom mohli označit například úlohu určování pohlaví člověka z fotografie, kde muž a žena představují dvě rozeznávané třídy. Předpovídání třídy ze vstupních dat potom nazveme klasifikací vstupních dat. V jádru jsou klasifikační úlohy podobné regresním úlohám, jelikož jednotlivé třídy popisujeme kvantitativně a jednotlivým třídám přiřazujeme hodnotu pravděpodobnosti. Třída s nejvyšší pravděpodobností se potom stává výstupem. Pro úlohy klasifikace se typicky používá učení s učitelem.[51]

Clustering, neboli shlukování je typ úlohy při které hledáme vztahy mezi vstupními daty a v závislosti na nich data rozdělujeme do skupin se společnými vlastnostmi. Pro tento typ úlohy se často používá učení bez učitele, typicky v případech kde vztahy mezi vstupními daty nejsou známy.[51]

## 3. Volně dostupné nástroje pro umělou inteligenci

Tato kapitola představuje několik ze současně nejpoužívanějších AI nástrojů. Jejich výběr byl inspirován článkem na internetové stránce Datamation[1].

### 3.1. Caffe

Caffe je open source framework pro strojové učení, konkrétně se zaměřuje na tzv. deep learning a využívá k tomu výpočetní model neuronových sítí. Název pochází ze zkratky Convolutional Architecture for Fast Feature Embedding a jeho vývoj probíhá na Kalifornské univerzitě v Berkley od roku 2014. Caffe klade důraz na rychlost, výpočtovou efektivitu a modularitu. Tvůrci tvrdí, že Caffe patří mezi nejrychlejší implementace konvolučních neuronových sítí.[2]

Caffe podporuje rozhraní MATLAB a Python a operační systémy Windows, Linux a macOS, dále podporuje platformu CUDA a umožňuje tak provádět výpočty na vybraných GPU. Jelikož je nástroj Caffe napsaný v jazyce C++, je možné Caffe používat také s nízkoúrovňovým C++ API, které je ale méně uživatelsky přívětivé. Již zmíněné rozhraní pro MATLAB a Python přidávají Caffe na funkcionalitě, ale k vyžívání tohoto open-source nástroje nejsou nezbytné. Caffe lze využívat i bez psaní jakéhokoliv kódu, například pouze s voláním základních funkcí s potřebnými parametry z příkazového řádku.[3]

Mezi základní parametry, které uživatel specifikuje pro učení neuronové sítě je tzv. model a solver, oba ve formátu prototxt. Parametr model definuje neuronovou síť, konkrétně má uživatel možnost určit počet a typ vrstev v síti a dále např. formát vstupních a výstupních dat. Parametr solver, neboli řešič, definuje jakým způsobem se bude neuronová síť učit, uživatel například určí, počet iterací, koeficient rychlosti učení, množství dat zpracovaných v jednom cyklu a další. Poslední potřebný parametr k učení sítě je dataset, ze kterého se bude síť učit.[3]

Procesem učení Caffe vyprodukuje tzv. caffemodel soubor, ve kterém jsou uloženy koeficienty „vycvičené“ sítě. Jelikož učení neuronové sítě je často velice výkonově a časově náročné, jsou volně k dispozici již vycvičené modely pro nejrůznější aplikace v tzv. Caffe Model Zoo.[2]

K práci s nástrojem Caffe jsou potřeba nástroje Make nebo CMake pro zkompilování a build programu. K dispozici je z oficiálních stránek i několik sestavených binárních souborů, které ale nepodporují všechny funkce Caffe, jako například podporu prostředí MATLAB. Podpory prostředí Python, MATLAB, nebo CUDA, je možné dosáhnout změnou nastavení při sestavování nástroje Caffe.[4]



Obrázek 3.1: Caffe logo [2]

## 3.2. Deeplearning4j

Deeplearning4j je open source deep learning knihovna určená pro jazyk Java a JVM jazyky. Knihovna Deeplearning4j se zaměřuje na jednoduchost použití a uživatelskou přívětivost na úkor možnosti podrobnější konfigurace. Toto umožňuje uživateli velice rychlé prototypování.[5]

Deeplearning4j podporuje primárně jazyk Java, ale je kompatibilní se všemi JVM jazyky, jako například Scala nebo Clojure. Nástroj Keras je zde možné využívat jako Deeplearning4j API pro jazyk Python. Dále Deeplearning4j podporuje platformu CUDA pro provádění výpočtů na GPU. Deeplearning4j je kompatibilní s operačními systémy Windows, Linux, macOS a také Android, umožňující jako jeden z mála nástrojů umělé inteligence využití na mobilních zařízeních.[7]

Deeplearning4j také nabízí rozsáhlé vizualizační uživatelské rozhraní, pomocí kterého je možné v reálném čase sledovat stav neuronové sítě, dále je možné zobrazit statistiky průběhu učení, což je používáno především k ladění parametrů učení.[8]

Pomocí Deeplearning4j je možné pracovat s neuronovými sítěmi základních typů, ale také vytvářet komplexní sítě spojováním sítí různých typů (jako například konvolučních nebo rekurentních sítí) do vrstev. Toto umožňuje uživateli vytvořit neuronovou síť „na míru“ konkrétní aplikaci.[7]



Obrázek 3.2: Deeplearning4j logo [5]

Pomocí nástroje Keras je možné do Deeplearning4j importovat modely dalších hlavních AI nástrojů, jako například Caffe, TensorFlow nebo Theano modely. Kromě této možnosti existují i volně dostupné modely přímo pro Deeplearning4j.[5]

Nástroje potřebné k práci s touto knihovnou jsou JDK (64-Bit), Apache Maven pro správu a řízení buildu, vývojové prostředí IntelliJ IDEA (nebo Eclipse) a Git.[7][6]

## 3.3. H2O

H2O je open source platforma pro machine-learning, která je vyvíjena firmou H2O.ai od roku 2011. Kromě výzkumné činnosti je H2O často používáno společnostmi pro vytváření prediktivních modelů, rizikové analýzy, analýzy pojištění a nebo například pro vývoj reklamních technologií. H2O je populární díky své rychlosti, jednoduchosti a množství algoritmů učení neuronových sítí.[9]

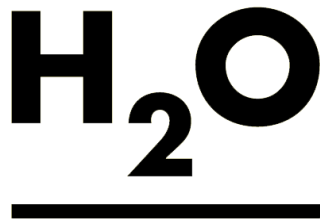
H2O podporuje jazyky Java, Scala, R a Python, z čehož Java je vyžadovaná, konkrétně JDK(64-Bit) které slouží k buildu H2O a jeho funkci. Z operačních systémů je možné H2O provozovat na Windows, macOS a Linux. Jako většina ostatních nástrojů svého typu podporuje platformu CUDA. V kombinaci s nástrojem Hadoop je možné s H2O vykonávat výpočty v tzv. clusteru, neboli paralelizovat výpočet na síti mnoha počítačů. Toto je vhodné řešení pro aplikace náročné na výkon. Dále je možné zkombinovat H2O

### 3.4. MICROSOFT COGNITIVE TOOLKIT (CNTK)

s nástrojem Spark, tato kombinace se nazývá Sparkling Water a umožňuje uživatelům používat pokročilé algoritmy strojového učení z prostředí Spark.[9]

H2O nabízí webové grafické uživatelské rozhraní H2O Flow, které je podporováno prohlížeči Chrome, Firefox, Safari a Internet Explorer. Pomocí tohoto rozhraní je možné například interaktivně pracovat s daty, vyvíjet nové modely nebo zobrazovat statistiky a výsledky učení neuronových sítí.[11]

Pro nástroj H2O je k dispozici škála modelů, ale i mnoho typů algoritmů pro učení neuronových sítí, které jsou rozděleny do dvou hlavních skupin, učení s učitelem a bez učitele. Všechny poskytované algoritmy jsou popsány na webové stránce H2O, což umožňuje uživateli jednoduchou volbu nejvhodnějšího algoritmu pro danou aplikaci.[12]



Obrázek 3.3: H2O logo [9]

## 3.4. Microsoft Cognitive Toolkit (CNTK)

Microsoft Cognitive Toolkit, původně známý pod zkratkou CNTK, je open source framework pro deep learning na komerční úrovni. Tento AI nástroj je vyvíjen společností Microsoft od roku 2014. Původně byl vytvořen pro účely společnosti Microsoft, ta z něj později kvůli úspěšnosti tohoto nástroje udělala open source projekt. Mnoho Microsoft služeb stále spoléhá na Microsoft Cognitive Toolkit.[14]

Microsoft Cognitive Toolkit lze používat jako knihovnu s jazyky Python, C# nebo C++, dále je možné Microsoft Cognitive Toolkit používat jako samostatný machine learning program s pomocí jazyka BrainScript. Podporované operační systémy jsou Windows a Linux. S Microsoft Cognitive Toolkit je možné provádět učení neuronových sítí na GPU a také tento nástroj umožňuje tzv. distribuované učení, tj. provádění učení na více počítačích.[14][13]

Microsoft Cognitive Toolkit je univerzální nástroj, který je možný použít pro řadu druhů aplikací, nejčastější typ využití je ale rozpoznávání řeči, obrazu a textu. Pro tyto účely jsou volně dostupné „vycvičené“ modely, ale i vhodné neuronové sítě pro tyto aplikace. Uživatel ale není limitován na sítě poskytované s Microsoft Cognitive Toolkit, je možné definovat vlastní sítě, popřípadě přizpůsobit již existující.[14][13]

Jako jeden z prvních deep learning nástrojů podporuje Microsoft Cognitive Toolkit formát Open Neural Network Exchange (ONNX). Jedná se o sdílený open source formát modelů vytvořený za účelem sdílení modelů a interoperabilitu deep learning frameworků. Díky tomuto nástroji je tedy možné sdílet modely například mezi nástroji Caffe2, MXNet, nebo PyTorch.[14]

Pro práci s Microsoft Cognitive Toolkit je požadováno vývojové prostředí preferovaného jazyka a instalace odpovídající verze Microsoft Cognitive Toolkit. Instalační soubory

jsou oficiálními zdroji poskytovány ve formě zkompileovaných binárních souborů, není tedy nutné nástroj kompilovat.[15]

## 3.5. Theano

Theano je open source Python knihovna pro numerické výpočty vyvinutá primárně za účelem machine learningu. Byla vyvíjena od roku 2009 na Universitě de Montréal a na konci roku 2017 tým vývojářů oznámil konec dalšího vývoje této knihovny (od verze Theano 1.0). [16]

Knihovna Theano je určená pro jazyk Python, je dostupná pro Windows, Linux a macOS, dále podporuje platformu CUDA pro vykonávání výpočtů na GPU.[16]

Hlavní vlastnost knihovny Theano je možnost definovat, optimalizovat a evaluovat matematické výrazy, předně maticové výrazy a také pracovat se symbolickými proměnnými. Další důležitá část této knihovny je optimalizační kompilátor, který kompiluje a dále zjednodušuje kód pro CPU a GPU. Díky optimalizačnímu kompilátoru je knihovna Theano schopná provádět náročné výpočty mnohem efektivněji a rychleji oproti vykonávání kódu pomocí Python interpretu. Zvláště ovlivněné jsou výpočty, které jsou opakovaně vyčíslovány, jako například výpočty při učení neuronových sítí.[16]

Theano je expresivní nástroj a uživatel je pomocí něj schopen definovat neuronové sítě a jejich učení na nízké úrovni. Oproti nástrojům jako například Caffé je tedy uživatel nucen konfigurovat většinu procesu „manuálně“ pomocí kódu, což umožňuje uživateli větší kontrolu nad vytvářeným AI aplikací. Nevýhodou je potom méně uživatelsky přívětivý vývoj aplikací machine learningu. Tuto nevýhodu řeší nástroj Keras, který lze používat jako nástavbu na Theano. Keras umožňuje s touto knihovnou pracovat pohodlněji, s jednodušší syntaxí na úkor expresivnosti.[17]



Obrázek 3.4: Theano logo [16]

Nástroje požadované k práci s knihovnou Theano jsou: Python, modul NumPy pro maticové výpočty, modul SciPy pro speciální funkce a BLAS. Theano podporuje instalaci těchto požadovaných nástrojů pomocí package manageru Conda. Další způsob instalace nástroje Theano a požadovaných nástrojů je package manager pip. Pro funkce jako například podpora kompilace kódu pro GPU nebo vytváření LaTeX dokumentace a další je nutné přidat další open-source nástroje.[17]

## 3.6. Mahout

Mahout je open source machine learning framework od společnosti Apache. Jeho vývoj začal v roce 2008. Tři hlavní využití tohoto nástroje jsou tzv. collaborative filtering, clustering a klasifikace, využívají jej především společnosti za účelem zpracování velkého množství dat.[19]

### 3.7. MLIB

Mahout podporuje programovací jazyky Java a Scala a operační systémy Windows, Linux a macOS. Tento nástroj je možné používat samostatně s jeho shell rozhraním, pomocí příkazového řádku, nebo jako knihovnu pro již zmíněné jazyky Java a Scala. Mahout také podporuje platformu CUDA. Mahout je dále kompatibilní s několika distributivními back-endy, doporučovaný je Apache Spark, který umožňuje Mahout aplikacím provádět distribuované výpočty. H2O je příklad dalšího kompatibilního back-endu.[20]

Mahout nabízí řadu machine learning algoritmů, nejčastější využitím jsou zmíněné collaborative filtering, clustering a klasifikace. Collaborative filtering je metoda, která například umožňuje předpovídat preference uživatelů na základě jiných uživatelů s podobnými preferencemi. Pro tento účel používají Mahout například společnosti Netflix nebo Pandora. Clustering je metoda řazení souboru dat do skupin na základě společných prvků. Algoritmus klasifikace nástroje Mahout je používán například pro tvorbu spam filtrů. Pro tyto účely jsou veřejně dostupné modely a také oficiální ukázky kódu pro vytváření aplikací pro zmíněné účely.[19]

Pro práci s Apache Mahout je vyžadováno Java JDK a dále Apache Maven pro sestavení nástroje Mahout.[19]

### 3.7. MLib

MLlib je machine learning knihovna pro cluster computing nástroj Spark od společnosti Apache. Pomocí nástroje MLib je možné s Apache Spark vytvářet machine learning aplikace pro tzv. big data. Jeho hlavní výhodou je vysoká rychlost machine learning algoritmů, v porovnání se starší metodou je MapReduce pro zpracování big data je přibližně 100 krát rychlejší.[22]

MLlib je možné používat s jazyky Java, Scala, Python a R. Dále je kompatibilní s platformou CUDA pro vykonávání učení neuronových sítí na CUDA kompatibilních grafických kartách. MLib je možné využívat skrze Spark API a dále je kompatibilní s Python Modulem NumPy nebo s knihovnami jazyku R.[22]

Knihovna MLib obsahuje řadu machine learning algoritmů pro regresi, klasifikaci, rozhodovací stromy, collaborative filtering, clustering a mnohé další. Dále poskytuje například také nástroje pro tvoření statistických aplikací.[22]

Prerekvizita pro práci s MLib je Apache Spark. Tento nástroj vyžaduje prostředí Java JDK a Scala. Dále je volitelně možné nainstalovat prostředí pro preferovaný jazyk, například Python.[22]

### 3.8. DLib

Dlib je general purpose open source knihovna. Od začátku svého vývoje v roce 2002 se postupně rozšiřovala o řadu nástrojů, jako například nástroje pro networking, vlákna, grafické rozhraní, komplexní datové struktury, lineární algebru, statistický machine learning, zpracování obrazu, data mining, XML, numerickou optimalizaci, Bayesovské sítě a mnohé další. V posledních letech se vývoj zaměřuje na vytváření souboru nástrojů pro machine learning, ale i přes to zůstává Dlib general purpose knihovnou.[23]

Knihovna Dlib je napsaná primárně pro jazyk C++ a je crossplatformní. Kromě C++ je možné knihovnu Dlib používat s jazykem Python pomocí Python API. Její funkčnost byla dle tvůrce ověřena u operačních systémů Windows, Linux, macOS a několika dalších,

### 3. VOLNĚ DOSTUPNÉ NÁSTROJE PRO UMĚLOU INTELIGENCI

méně známých operačních systémů. Záměr vývojářů při tvorbě knihovny Dlib je portabilita a jednoduchost použití, tak aby uživatel nebyl nucen složitě knihovnu konfigurovat. Také Dlib podporuje platformu CUDA.[23][27]

V oblasti machine learningu obsahuje Dlib širokou řadu základních algoritmů, jako například algoritmy pro klasifikaci, clustering, regresi a další. Všechny funkce knihovny jsou podrobně zdokumentovány a je poskytnut i stručný návod k výběru vhodné funkce pro daný typ aplikace. Tvůrce knihovny, Davis King, poskytuje s knihovnou i řadu vycvičených modelů pro účely machine learning aplikací, spolu s ukázkovým kódem, kde demonstrovuje jejich použití. Zahrnuty jsou například modely pro rozpoznání obličeje nebo auta.[24][26]

Před použitím je nutné zdrojový kód knihovny Dlib zkompileovat, jelikož oficiální zdroje nenabízí zkompileované binární soubory k instalaci. K tomu je doporučený nástroj CMake a Visual Studio, během tohoto procesu může uživatel specifikovat kompilaci rozhraní Python pro použití s DLib, kromě výchozího C++ API. V případě práce v prostředí Python je další možnost instalace package manager pip.[25]



Obrázek 3.5: DLib logo [23]

### 3.9. NuPIC

NuPIC je open source nástroj používající machine learning algoritmy založené na modelech mozkové kůry. NuPIC, neboli Numenta Platform for Intelligent Computing, je vyvíjený společností Numenta, která v roce 2013 udělala tento projekt open source.[28]

Nástroj NuPIC je možné používat s jazyky C++ a Python. Dále existují i nástroje používající stejné algoritmy machine learningu od společnosti Numenta, jako HTM.java pro jazyk Java a Comportex pro jazyk Clojure. Z operačních systémů podporuje NuPIC Windows, Linux a macOS. NuPIC nabízí 2 druhy API, Network API a Online Prediction Framework API. Network API je primární, nízkourovňové API, které nabízí uživateli vyšší stupeň konfigurace sítí. API Online Prediction Framework, nebo OPF, je vysokoúrovňové Python API zjednodušující práci s nejčastěji používanými modely. OPF je zaměřené na experimentaci s předpovědmi, detekci anomálií a na tzv. Swarming.[29]

Machine learning algoritmy nástroje NuPIC jsou implementací teorie HTM, neboli Hierarchical Temporal Memory. HTM je machine learning technologie, která se snaží zachytit strukturální a algoritmické vlastnosti mozkové kůry. HTM je druh neuronové sítě, se specifickým hierarchickým uspořádáním neuronů do sloupců, vrstev a regionů. Na rozdíl od jiných machine learning algoritmů se učí HTM sítě na časově závislých neoznačených souborech dat a spoléhají na ukládání velkého množství vzorků a sekvencí dat.[30]

## 3.10. TensorFlow

TensorFlow je open source machine learning software od společnosti Google. Jedná se momentálně o jeden z nejpobulárnějších AI nástrojů, který je používáný mnoha velkými společnostmi jako například Google, Uber, Ebay nebo Airbnb. TensorFlow byl vytvořen skupinou Google Brain team pro účely výzkumu deep learning neuronových sítí a je nástupcem Google projektu DistBelief.[32]

TensorFlow nabízí API pro jazyky Python, C++, Java a Go. Rozhraní pro jazyk Python je nejúplnější, nejjednodušší na použití a také nejvíce používané. Dále jsou k dispozici API pro TensorFlow od vývojářů třetích stran pro jazyky C#, Haskell, Julia, Ruby, Rust a Scala. TensorFlow podporuje platformu a operační systémy Windows, Linux, macOS a Android.[33][34]

TensorFlow vykonává numerické výpočty pomocí diagramů datových toků. Uzly těchto diagramů reprezentují matematické operace, a spojnice mezi uzly reprezentují tenzory, které mezi nimi proudí. Název TensorFlow je inspirovaný tímto konceptem. Tento způsob vyjadřování machine learning modelů napsaný v C++ a CUDA umožňuje rychlé a efektivní vykonávání výpočtů učení neuronových sítí.[34]



Obrázek 3.6: Tensorflow logo [32]

Pro vizualizaci TensorFlow aplikací je k dispozici nástroj TensorBoard. Tento nástroj umožňuje zobrazit model, diagram jeho struktury, informace týkající se učení sítě a umožňuje uživateli ladit a optimalizovat TensorFlow aplikace. TensorBoard funguje na bázi webové aplikace a podporuje prohlížeče Google Chrome a Mozilla Firefox.[35]

Kromě samotného nástroje TensorFlow jsou od vývojářského týmu TensorFlow volně dostupné také oficiální modely a ukázky kódu demonstrující možnosti tohoto softwaru. Díky široké komunitě nástroje TensorFlow jsou dostupné také neoficiální modely a mnoho dalších zdrojů, především pro Python API.[35]

## 3.11. Torch

Torch je open source framework pro vědecké výpočetní účely s širokou podporou pro machine learning. Nástroj Torch je vyvíjen od roku 2002, kdy jeho vývojáři v tomto frameworku implementovali v té době nejmodernější machine learning algoritmy a udělali

### 3. VOLNĚ DOSTUPNÉ NÁSTROJE PRO UMĚLOU INTELIGENCI

jej veřejně dostupným vývojářské komunitě. Torch se zaměřuje na podporu vykonávání machine learning procesů pomocí GPU a podporuje jazyk LuaJIT.[36]

Nástroj Torch nabízí API pro jazyk LuaJIT, jednoduchý a rychlý skriptovací jazyk. Ohledně operačních systémů podporuje Torch systémy Windows, Linux, macOS a pro mobilní aplikace také Android a IOS. Dále Torch podporuje také platformu CUDA.[37]

Hlavní cíl nástroje Torch je maximální flexibilita, rychlost a jednoduchost při vytváření vědeckých algoritmů. Mezi hlavní vlastnosti machine learning frameworku Torch patří rychlé funkce pro numerické výpočty s n-rozměrnými poli, funkce lineární algebry, modely neuronových sítí, energy-based modely, funkce pro numerickou optimalizaci a efektivní podpora GPU. Nástroj Torch je určen pro aplikace machine learningu, počítačového vidění, zpracování signálů, obrazu nebo audia.[36]

Jazyk LuaJIT je interpretovaný skriptovací jazyk založený na jazyku Lua. Podporuje procedurální programování, objektově-orientované programování, funkcionální programování a další. Tento jazyk se zaměřuje na využití ve vestavěných systémech a hrách a je nejvyužívanějším skriptovacím jazykem v oblasti her. Hlavní výhody jazyku Lua je rychlost, přenosnost a také open-source licence.[39]

Pro instalaci Torch jsou dostupné neoficiální sestavené binární soubory, primární způsob instalace je ale pomocí nástroje CMake, kdy si uživatel musí ze zdrojového kódu Torch vytvořit vlastní build. Nejpodporovanější platforma a nejjednodušší pro instalaci je Linux, pro jiné operační systémy je tedy alternativou k instalaci práce s nástrojem Torch na systému Linux na virtuálním stroji.[38]

## 4. Volně dostupná data pro učení neuronových sítí

Pro vytvoření modelu umělé inteligence založeného na principu neuronových sítí je zapotřebí provést jeho učení pomocí tzv. datasetu sestaveného pro danou aplikaci. Dataset pro učení neuronových sítí je soubor dat, který obsahuje možné vstupy a v případě učení s učitelem, i pro ně požadované výstupy. Prostřednictvím vhodného algoritmu učení se dokáže model umělé inteligence adaptovat pro vyhodnocování vstupů způsobem napodobující data v datasetu. Může se jednat o data v jakékoliv formě, například grafická, audio, nebo numerická data, u kterých se manuálně označí sledované vlastnosti. Velikost datasetů se v závislosti na typu dat a složitosti aplikace pohybuje nejčastěji v desítkách tisíců až v desítkách milionů vzorků dat u některých případech. Pro vytvoření robustního modelu umělé inteligence je nezbytný rozsáhlý a rozmanitý dataset, nevýhoda takovýchto objemných datasetů je obtížnost jejich sestavení, označení a dále ze své podstaty delší doba učení modelu.[51]

Podobně jako volně dostupné nástroje pro vytváření modelů umělé inteligence existuje i množství volně dostupných dat pro jejich učení. Jelikož sestavování datasetů pro účely machine learning aplikací je časově náročné, vzniklo několik platforem pro jejich sdílení. Kromě zdrojů zaměřených přímo na machine learning datasety je možné získat data pro učení neuronových sítí z jiných veřejných zdrojů.

### 4.1. Kaggle

Kaggle je jednou z nejpoblárnějších platforem pro sdílení dat určených k vytváření prediktivních AI modelů a obecně k analýze dat. Volně k dispozici je přes 12 000 datasetů různých zaměření. Datasety jsou ve formátech CSV, SQLite, JSON, nebo BigQuery.[40]

Důležitý prvek platformy Kaggle jsou soutěže ve vytváření prediktivních AI modelů. Vypisovatel soutěže na platformě Kaggle poskytne zadání, dataset pro daný úkol a v některých případech také finanční odměnu pro výherce. Zapojit do soutěže se může každý po registraci, stejně tak může i každý soutěž vypsat. Možnosti vypsat soutěž na platformě Kaggle využívají i společnosti, které za vítězné řešení úkolů, například z oblasti biologie, ekonomiky nebo sportu, vyplácejí odměny až v rádech milionů USD. Přednost řešení problémů pomocí soutěže je velké množství a tudíž rozmanitost způsobů řešení, jelikož u prediktivního modelování není předem jednoznačné, která metoda řešení je nejefektivnější.[40]



Obrázek 4.1: Kaggle logo [40]

## 4.2. UCI Machine Learning Repository

UCI Machine Learning Repository nabízí datasety pro machine learning a také generátory dat pro empirickou analýzu machine learning algoritmů. UCI Machine Learning Repository je využíváno studenty, pedagogy a výzkumníky a za více než 30 let své existence bylo citováno ve více než 1000 pracích. Celkově nabízí UCI Machine Learning Repository přes 400 datasetů pro účely klasifikačních, regresních, clustering a dalších typů aplikací.[41]

## 4.3. Další zdroje volně dostupných dat pro machine learning

Kromě rozsáhlých a populárních stránek pro sdílení machine learning dat jako Kaggle a UCI Machine Learning Repository existuje i řada menších webových úložišť pro tento účel.

Mldata.org, neboli machine learning dataset repository, je stránka zaměřující se na sdílení materiálů pro machine learning. Jedná se o alternativu k platformě Kaggle, fungující na podobném principu. Uživatelé na této stránce mohou sdílet jak datasety, tak i algoritmy pro jejich využívání v machine learning aplikacích. Dále je na mldata.org možné vytvářet soutěže vytváření AI modelů, podobně jako na Kaggle.com.[42]

Další zdroj volně dostupných dat je [aws.amazon.com/datasets/](https://aws.amazon.com/datasets/). Zde služba AWS poskytuje volně k dispozici datasety různého zaměření, například enviromentální, finanční, nebo data z oblasti genetiky.[43]

Google BigQuery je služba pro zpracování velkých množství dat. V rámci této služby Google také poskytuje volně dostupná data, se kterými mohou uživatelé pracovat pomocí SQL příkazů z webového rozhraní, nebo z API pro jazyky Java, .NET, nebo Python. Uživatelé mohou zdarma zpracovat 1 Terabyte dat za měsíc pomocí služby BigQuery.[45]

Microsoft Azure poskytuje volně dostupné datasety z různých oblastí, převážně se ale jedná o statistická data.[44]

Quandl.com je zdroj dat primárně z finanční a ekonomické oblasti. Část nabízených dat je volně k dispozici. Mezi typy dat, které zde najdeme patří například cenové vývoje komodit, akcií, a jiné statistiky. Jedná se o data, která jsou v oblasti machine learningu používána předně pro vývoj prediktivních modelů, určených pro ekonomické předpovědi. Podobně jako Quandl.com, [datacatalog.worldbank.org](https://datacatalog.worldbank.org/) poskytuje data týkající se ekonomiky a finančnictví. Kromě těchto oblastí je zde možné najít také geografická, nebo statistická data.[46][47]

## 5. Ukázky vybraných nástrojů

### 5.1. Caffe

Ukázka nástroje Caffe (příloha 1) je jednoduchá klasifikační úloha. Jedná se klasifikaci obrázků psů a koček, model umělé inteligence má tedy 1 vstup ve formě obrázku a výstupem je jedna ze dvou tříd. Je nutné vytvořit tři soubory, jeden definující umělou neuronovou síť pro proces učení, druhý definující umělou neuronovou síť pro finální použití a třetí definující řešič, tj. jakým způsobem se bude model učit. Jako neuronová síť byla zvolena síť AlexNet, jedna z ukázkových sítí, která je součástí softwaru Caffe. Jedná se o konvoluční neuronovou síť určenou pro zpracování obrazu. Neuronová síť je definovaná v souborech ve formátu prototxt, ve kterém je nutné nastavit příslušný tvar vstupu. Soubor řešiče je taktéž ve formátu prototxt, v tomto souboru se nastavují tzv. hyperparametry, parametry popisující učení, data pro učení a také cesta k souboru s neuronovou sítí, která byla zvolena. Dalším krokem jsou data. Pro tuto úlohu byl použit volně přístupný dataset z platformy Kaggle obsahující 25 000 obrázků psů a koček v poměru přibližně 1:1. Pro rychlejší přístup k datům byl z datasetu vytvořen databázový soubor ve formátu mdb pomocí nástroje `convert_imageset.exe`, který je součástí softwaru Caffe. Caffe podporuje několik programovacích jazyků, úlohu tohoto typu je však možné řešit pomocí příkazového řádku. Pro spuštění procesu učení byl zadán do příkazového řádku příkaz

```
>>caffe train -solver solver.prototxt
```

kde soubor `solver.prototxt` je zmíněný řešič. Proces učení probíhá počet iterací stanovený v souboru `solver.prototxt` a po dokončení vyprodukuje soubor typu `caffemodel`. Tento soubor udává koeficienty pro neuronovou síť, která byla zvolena a dohromady tvoří výsledný model. Model lze použít s příkazem `test`, například

```
>>caffe test -model alexnet_train.prototxt -weights caffe_alexnet_train_iter_80000.caffemodel -iterations 100
```

Tento příkaz vyhodnotí model na testovacím datasetu, souboru obrázků, které nebyly použity k učení a určí procentuální úspěšnost. Po 80000 iteracích učení byla dosažena přesnost klasifikace 92.725%.

```
21:02:11.179844   Iteration 80000, Testing net (#0)
21:02:40.157727   Test net output #0: accuracy = 0.92725
21:02:40.157727   Test net output #1: loss = 0.235957 (* 1 = 0.235957 loss)
```

### 5.2. Tensorflow

Pomocí nástroje Tensorflow byla vytvořena implementace modelu umělé inteligence k rozeznávání objektů (příloha 2). Jedná se o volně dostupný model [60] od vývojářského týmu Tensorflow vycvičený k rozeznávání 1 000 tříd objektů na datasetu ImageNet, obsahujícím přes 14 000 000 obrázků. Vzhledem k velikosti datasetu není praktické vytvářet vlastní model na běžném hardwaru, byl tedy použit již zmíněný vycvičený volně dostupný model založený na neuronové síti InceptionResNet V3 a spolu s ním byla použita

## 5. UKÁZKY VYBRANÝCH NÁSTROJŮ

jeho funkce na vyhodnocování předpovídané třídy `classify_image`. Cílem této ukázky je použít model na rozeznávání objektů v reálném čase. Ke snímání obrazu byla použita knihovna OpenCV, která umožňuje snímání videa webkamerou. Snímaný obraz je dále pomocí funkce `classify_image` poslán do zmíněného modelu, zpracován a vrácena je hodnota nejpravděpodobnější třídy. Jelikož zpracování obrazu trvá v porovnání s periodou videa relativně dlouho je nutné proces snímání videa a proces zpracování obrazu pomocí modelu umělé inteligence rozdělit do paralelních procesů v samostatných vláknech. Výsledek zpracování, nejpravděpodobnější třída, je zobrazena v horní části obrazu, vedle třídy je zobrazena jistota (score) správného určení třídy. Na obrázku 5.1 je možné vidět ukázku správné klasifikace 4 předmětů.



Obrázek 5.1: Rozpoznání předmětů pomocí Tensorflow aplikace

### 5.3. Dlib

Další ukázka (příloha 3) je vytvořena s knihovnou DLib a programovacím jazykem Python. Knihovna DLib byla využita v programu na stabilizaci a výřez obličeje z obrázku. K detekci byla použita funkce `frontal_face_detector`, která přibližně lokalizuje obličeje v obrázku. Polohy obličejů jsou dále předány funkci `shape_predictor`, která lokalizuje obličejové prvky jako oči, nos, obočí a ústa. Pomocí těchto prvků, referenčních bodů obličeje, dokáže funkce `FaceAligner` natočit obličej do správné polohy a umístit jej do středu výřezu. V posledním kroku je změněna velikost výřezu na požadovanou velikost. Výhoda

## 5.4. THEANO

používání DLib funkcí k lokalizaci a stabilizaci obličeje je robustnost. Funkce knihovny DLib dokáže detekovat i obličeje natočené v nepříznivých úhlech a dále je, v případě potřeby, spolehlivě stabilizovat. Nevýhodou je vyšší výpočetní náročnost při detekci. Na obrázku 5.2 je možné vidět ukázkou výřezu a stabilizace obličeje.



(a) Původní obrázek



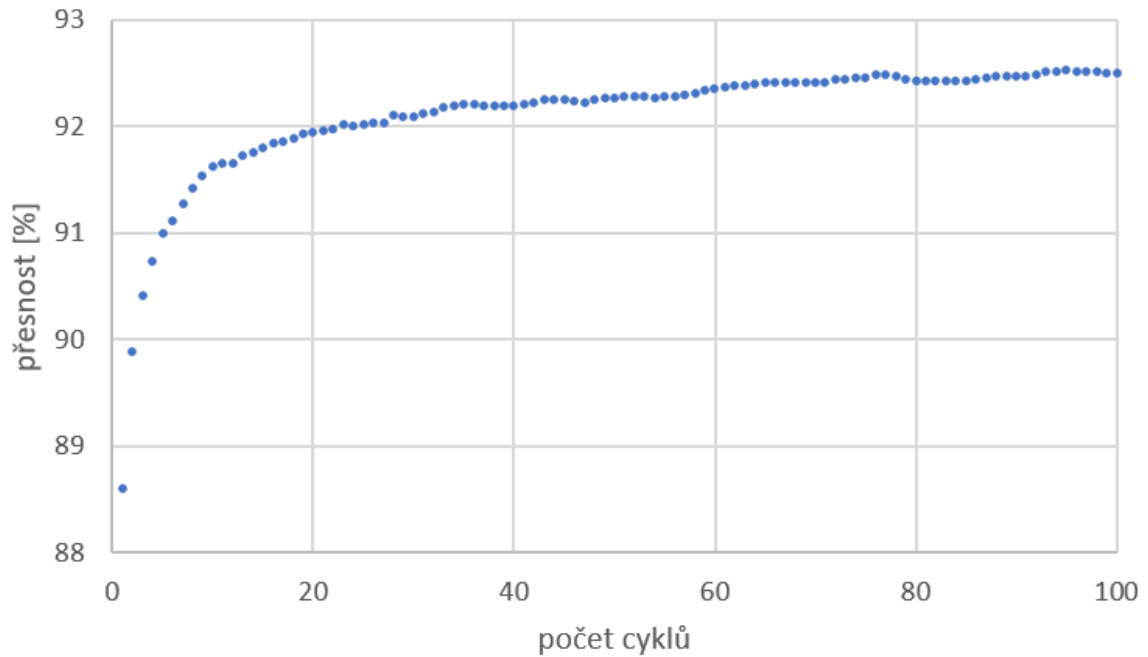
(b) Přibližný výřez obličeje      (c) Stabilizovaný obličej

Obrázek 5.2: Stabilizace obličeje pomocí knihovny DLib

## 5.4. Theano

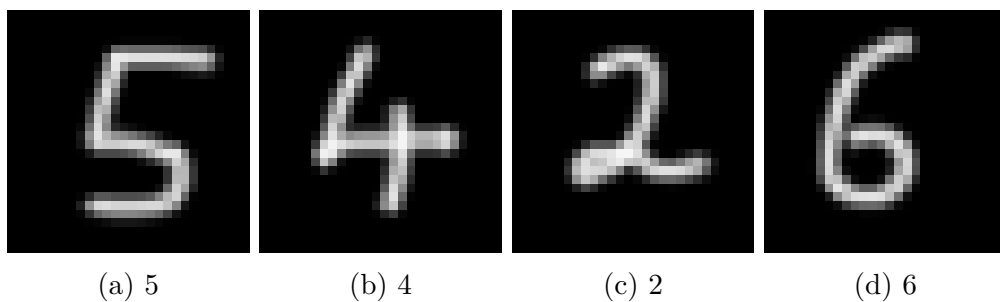
V ukázce nástroje Theano (příloha 4) byla použita metoda logistické regrese ke klasifikování rukou napsaných číslic, jedná se tedy o klasifikační úlohu o 10 třídách. Model byl vycvičen na datasetu MNIST [59], obsahujícím 60 000 rukou napsaných číslic ve formátu 28 na 28 pixelů pro učení a 10 000 číslic pro ověřování přesnosti během učení. Je důležité, aby část datasetu určená k ověřování přesnosti nebyla používána k učení, v takovém případě by model mohl vykazovat nerealistickou přesnost. Pro zvýšení rychlosti zpracování dat se obrázky převedou na jednorozměrné transformací z matice 28x28 na matici 728x1, tedy na vektor.

Po jednom cyklu učení je přesnost 88.6%, po 100 cyklech se přesnost klasifikace zvýšila na 92.5%. Z grafu závislosti učení (obrázek 5.3) je zřejmé, že hodnota přesnosti konverguje přibližně k hodnotě 93%, při dalších cyklech učení by tedy už nedošlo k výraznému nárůstu přesnosti.



Obrázek 5.3: Závislost přesnosti klasifikace na počtu cyklů učení

Po dokončení procesu učení aplikujeme model pro demonstraci na 4 vlastní číslice (obrázek 5.4).



Obrázek 5.4: Klasifikované obrázky číslic

```
>>predicted = predictor(img_flat)
>>print(predicted)
[5,4,2,6]
```

U všech 4 číslic model dokázal správně klasifikovat třídu.

## 6. Aplikace

Pro ukázkou rozsáhlejšího použití některých z popsaných nástrojů byla vytvořena aplikace umělé inteligence demonstrující jejich možnosti (příloha 5).

### 6.1. Volba prostředí a nástrojů

Při volbě prostředí pro tvoření aplikace bylo bráno v potaz několik požadavků, jako například podpora programovacích jazyků, operační systém, uživatelská přívětivost, rychlost a také dostupnost zdrojů pro daný software. S ohledem na tyto požadavky byl zvolen open source nástroj TensorFlow, který podporuje širokou řadu programovacích jazyků, je multiplatformní, rychlý a předně, díky své popularitě a rozsáhlé komunitě, má oproti ostatním nástrojům velké množství veřejně dostupných zdrojů, které zjednodušují vývoj aplikací na této platformě. Mezi tyto zdroje patří rozsáhlá dokumentace, množství veřejně přístupného kódu, AI modelů, neuronových sítí, nebo online komunity zabývající řešením častých problémů.

Python byl zvolen jako programovací jazyk pro svoji univerzálnost a kompatibilitu s nástroji, které byly v tomto projektu použity. Jednou z výhod je také uživatelská přívětivost, díky jednoduchosti syntaxe. Dále z pohledu nástroje TensorFlow je API pro jazyk Python nejvíce podporované aplikační rozhraní. Jako vysokoúrovňový interpretovaný jazyk má Python nevýhodu v nižší rychlosti oproti kompilovaným programovacím jazykům, jako například jazyk C++, toto však nehraje velkou roli, jelikož Python pouze komunikuje s API a většina výpočetního výkonu je soustředěna na procesy nástroje TensorFlow. Použití kompilovaného jazyku by tedy sice mělo přispět k urychlení celé aplikace, ale ne výrazně. V případě této aplikace, která není velmi náročná na výkon, není tedy nutné používat kompilovaný jazyk z tohoto důvodu.

Dalším nástrojem použitým v této aplikaci je open source knihovna OpenCV, neboli Open Source Computer Vision Library. OpenCV je knihovna pro účely počítačového vidění pro jazyky Python, C++ a Java. OpenCV se zaměřuje na efektivní zpracování obrazu v reálném čase, což je jeden z požadavků vyvíjené aplikace. Dále OpenCV nabízí nástroje pro efektivní detekci objektů pomocí metody Viola-Jones.

Knihovna Dlib byla použita pro funkce z oblasti machine learningu a počítačového vidění. Konkrétně pro funkci rozpoznání obličejových prvků, která je použita v této aplikaci pro předběžné zpracování obrazu. Použití funkcí z knihovny Dlib je jednoduché a uživatelsky přívětivé a bylo by možné knihovnu Dlib použít pro více účelů v celé aplikaci, ne všechny funkce jsou ale vhodné pro zpracování obrazu v reálném čase z důvodu přílišné náročnosti na výpočetní výkon.

### 6.2. Funkcionalita

Úkolem aplikace je detekce obličejů snímaných webkamerou a jejich následné zapamatování. Aplikace je schopna známé obličeje (dříve viděné nebo z externě dodaných dat) identifikovat v reálném čase.

### 6.3. Struktura aplikace

V dané aplikaci jsou dva hlavní procesy, proces snímání a předběžného zpracování obrazu a dále proces zpracovávání dat pomocí AI modelu. Aplikace je určena pro zpracování dat v reálném čase, což přináší řadu komplikací, zejména při zachování konstantní a dostatečně vysoké frekvence snímaného obrazu. Jelikož je proces zpracovávání dat pomocí neuronové sítě poměrně časově náročný, musela být použita paralelizace těchto dvou procesů. Pro rozdělení procesů do dvou nezávislých vláken byl použit Python Multiprocessing modul, toto umožňuje zachovat stálou vyšší frekvenci videa. Frekvence videa závisí na konkrétním zařízení, na průměrném stroji se ale pohybuje frekvence videa kolem 20 snímků za sekundu.

Struktura aplikace vypadá následovně: po spuštění nejprve nastává inicializační fáze, ve které probíhá načítání modelu a externích obrázků pro pozdější porovnávání se snímaným obrazem. Z důvodu velikosti modelu může tato fáze trvat až několik minut. Po inicializační části probíhá proces ve dvou paralelizovaných while cyklech. V prvním while cyklu probíhá snímání obrazu webkamerou, což je uskutečněno pomocí knihovny OpenCV a dále předběžné zpracování obrazu. Funkce knihovny OpenCV společně s knihovnou Dlib zpracuje při každém cyklu snímaný obraz a vrátí výřezy obličejů, které jsou asynchronně předány do druhého while cyklu, který obsahuje AI model. Výsledek ze zpracování je asynchronně vrácen zpět do předchozího cyklu, kde proběhne porovnání s databází a vyhodnocení shody. Shoda se vyhodnocuje podle empiricky zvolené hranice podobnosti. Hodnoty pod touto hranicí jsou považovány za shodu, hodnoty přesahující tuto hranici značí rozdílné obličeje. Po vyhodnocení shody jsou výsledky promítnuty na snímaný obraz ve formě čtverců, značící detekované obličeje a jmen v levém horním rohu čtverců.

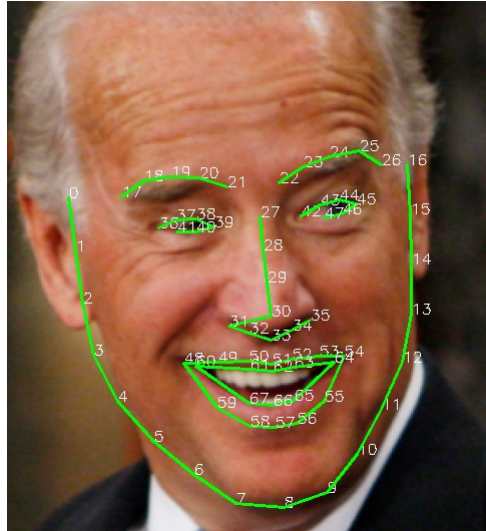
### 6.4. Předběžné zpracování obrazu

Obraz snímaný kamerou je nutné zpracovat a specifickým způsobem upravit před vstupem do neuronové sítě. Zejména je nutné detekovat obličeje přítomné v obrazu, izolovat je, změnit jejich velikost a natočit je do správné orientace. Výstupem je tedy výřez obličeje z původního obrazu o konkrétní velikosti, kde obličej ve vztahu k výřezu má konstantní velikost a polohu. Tohoto je dosaženo rozpoznáním obličejových prvků a vytvoření výřezu v závislosti na jejich poloze. Na závěr se na výsledný výřez aplikuje statistická funkce, pro vytvoření obrazu s nulovou střední hodnotou. Díky tomuto je obraz pro další zpracování více konzistentní a méně ovlivněný rozdíly v jasu.

K předběžnému zpracování obrazu jsou použity knihovny OpenCV a Dlib. Z knihovny OpenCV byla použita funkce pro detekci obličejů založená na metodě Viola-Jones. Tato funkce byla zvolena z důvodu malých nároků na výpočetní výkon, její výhodou je tedy vyšší rychlost. Nevýhodou této metody je menší robustnost – menší rozsah úhlů ve kterých dokáže detekovat obličej. Jelikož pro další zpracování je nutné zachytit obličej co možná s nejmenším natočením do stran, nejlépe hledící přímo, není z toho důvodu nezbytné používat robustnější metodu detekce. Tato metoda slouží pouze k předběžné lokalizaci obličeje, dále je nutné obličej přesněji zarovnat a také natočit kolem osy kolmé na rovinu obrazu. K tomuto je použita funkce z knihovny Dlib, která na obličej lokalizuje 68 bodů definující obličejové prvky, konkrétně oči, obočí, nos, ústa a čelist (obrázek 6.1). Pro otočení je použita poloha očí, kde žádané natočení je takové, kde příčka spojující oči je

## 6.5. MODEL

vodorovná. Poloha obličeje ve výřezu je upravena také v závislosti na poloze očí. Tímto způsobem je dosaženo konzistentní stabilizace obličejů přítomných v obraze a je umožněno další zpracování.



Obrázek 6.1: Detekce obličejových prvků pomocí DLib [48]

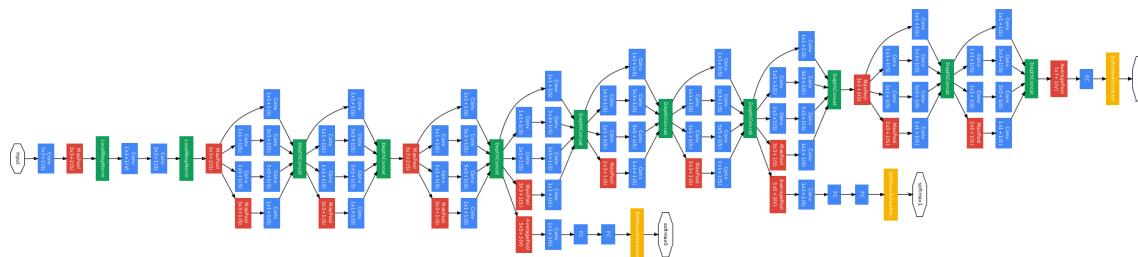
## 6.5. Model

Model je založený na algoritmu popsáném v „FaceNet: A Unified Embedding for Face Recognition and Clustering“. Hlavní myšlenka tohoto modelu spočívá v transformaci obrazu obličeje na tzv. embedding – objemově malou reprezentaci obličeje, ve formě  $x$ -dimenzionálního vektoru, definující daný obličej. Poloha vektoru v  $x$ -rozměrném prostoru přímo souvisí se vzhledem obličeje, podobnost obličejů se pak určuje na základě velikosti vektoru, který je rozdílem dvou zkoumaných vektorů. Ve své podstatě je podobnost obličejů úměrná vzdálenosti dvou poloh v prostoru. Čím jsou tedy vektory reprezentující obličej blíže k sobě, tím si jsou obličej podobnější. V aplikaci popisované zde je používán embedding o 512 dimenzích, s nárůstem počtu rozměrů roste i komplexita dat, které může embedding reprezentovat, nevýhodou je pak větší datový objem a větší nároky na výpočetní výkon.[56]

Výhodou transformace na embedding je vysoká efektivita v ohledu na poměr datového objemu a objemu dat které reprezentuje. Ve své podstatě se jedná o kompresi, kde cílem je eliminovat neesenciální data a zachovat pouze relevantní data charakterizující vzhled obličeje, datový objem se pak pohybuje v řádech stovek bytů na embedding, v závislosti na počtu rozměrů. Toto má za důsledek i výhodu ve vytváření databáze obličejů, je totiž možné ukládat už zpracovaná data v embedding formě, což drasticky snižuje nároky na prostor. Tato metoda ukládání dále šetří výpočetní výkon, jelikož v případě budoucího porovnávání se porovnává nově vytvořený embedding s již zpracovanými daty v databázi. Jak již bylo zmíněno, proces porovnávání pak tkví v pouze v rozdílu vektorů, což je poměrně jednoduchá a na výpočetní výkon nenáročná operace.[56]

Model je uskutečněný pomocí volně dostupné neuronové sítě Inception ResNet v1 (obrázek 6.2), která je někdy také označovaná jako GoogLeNet. Jedná se o konvoluční neuronovou síť určenou pro zpracování obrazu s relativně malými nároky na výpočetní

výkon. Inception ResNet v1 se stala vítězem ILSVRC2014 (Large Scale Visual Recognition Challenge 2014) – soutěže hodnotící algoritmy pro obrazovou klasifikaci.[49]



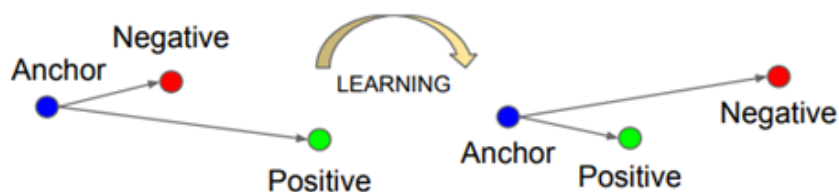
Obrázek 6.2: Schéma neuronové sítě Inception ResNet v1 [50]

## 6.6. Učení modelu, dataset

Z důvodu vysoké časové náročnosti učení vzhledem k vyžadované velikosti datasetu, byl pro aplikaci přejat volně dostupný model [57]. Tento model byl vycvičen pomocí implementace algoritmu popsaného v „FaceNet: A Unified Embedding for Face Recognition and Clustering“.

Cílem procesu učení u tohoto modelu je docílení závislosti u výstupu modelu (embedding) mezi vzhledem a pozicí v daném 512 rozměrném prostoru. Ve své podstatě se proces učení snaží docílit, aby výstup modelu – vektory reprezentující obličeje, byly polohou v prostoru blízko dalším vektorům, které reprezentují vzhledově podobné obličeje. V případě odlišných obličejů je příznivé, pokud vytvořený embedding je v prostoru co nejvíce vzdálený od porovnávaného.[56]

Algoritmus učení modelu je založený na metodě trojic. V každém cyklu učení je modelu dodána trojice vstupů – referenční, pozitivní a negativní. Referenční vstup je jakýkoliv obličej, pozitivní vstup je obličej stejné osoby jako referenční a negativní vstup je obličej jiné osoby. Provádí se vždy porovnání referenčního vstupu se dvěma zbylými vstupy. Při procesu učení se usiluje o maximalizaci vzdálenosti mezi výstupy odpovídající referenčnímu a negativnímu vstupu, a naopak minimalizaci vzdálenosti mezi výstupy odpovídající referenčnímu a pozitivnímu vstupu (obrázek 6.3). Při vytváření trojic pro učení modelu je vhodné vybrat negativní vstup vzhledově blízký referenčnímu, v opačném případě by rozlišování pozitivního vstupu od negativního bylo pro model „jednoduché“ a učení by tak nebylo dostatečně efektivní.[56]



Obrázek 6.3: Schéma algoritmu učení [56]

## 6.7. VYTVÁŘENÍ SEZNAMU ZNÁMÝCH OBLIČEJŮ

Jako dataset pro učení byl použit VGGFace2, jeden z největších veřejně dostupných obličejových datasetů obsahující přibližně 3 300 000 obrázků obličejů od 9 000 osob. Dataset pro tuto aplikaci musí obsahovat vždy více obrázků pro danou osobu, aby bylo možné vytvářet zmíněné trojice vstupů, to dataset VGGFace2 splňuje.[58]

## 6.7. Vytváření seznamu známých obličejů

Jelikož se jedná o aplikaci pro rozpoznávání obličejů, je nutné vytvořit seznam známých obličejů pro budoucí porovnávání a po čas běhu aplikace je třeba ukládat nové obličeje. Položku v seznamu známých obličejů tvoří embedding obličeje vytvořený AI modelem a odpovídající identifikátor daného obličeje ve formě textového řetězce. Ukládání již vytvořených embedding reprezentací obličejů je výhodnější z hlediska výpočetní náročnosti, jelikož embedding je možné přímo vyhodnocovat. Každý nový embedding je porovnáván s tímto seznamem a pokud není nalezena shoda, označuje se obličej za neznámý. V tomto případě je možné neznámý obličej přidat do seznamu pomocí stisku klávesy.

## 6.8. Načítání externích dat

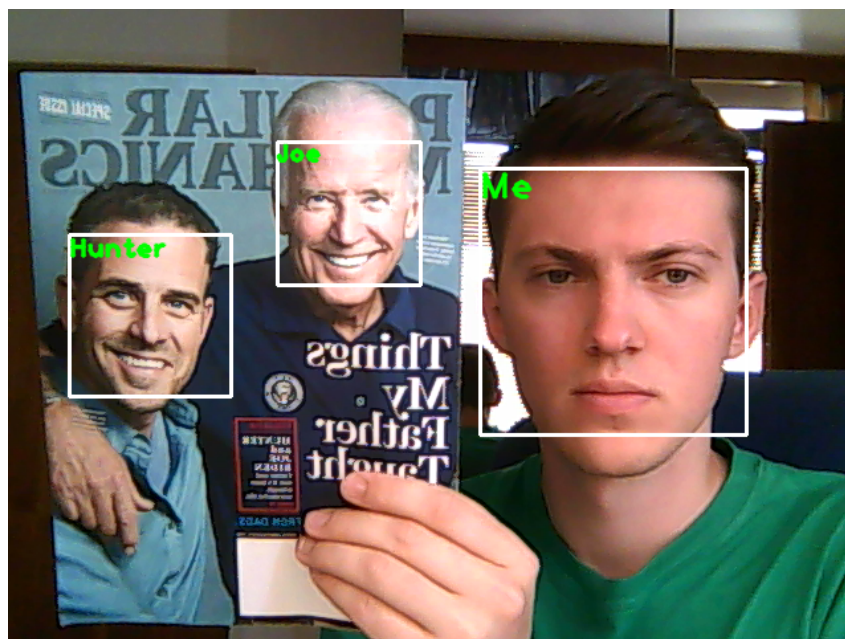
Aplikace je schopna načíst externí fotografie a pomocí těchto fotografií posléze identifikovat snímané osoby. V první části procesu jsou získány obrázky z předem určené složky, u kterých se předpokládá vždy jeden obličej na fotografii. Název souboru zde slouží jako identifikace dané osoby, například jméno. V dalším kroku je nutné z obrázků izolovat a stabilizovat obličeje pro pozdější zpracování, tohoto je dosaženo metodou popsanou v sekci „Předběžné zpracování obrazu“. Dále jsou připravené výřezy obličejů poslány ke zpracování AI modelem, který vrací jejich embedding tvar. Každý embedding je poté přidán do seznamu známých obličejů společně s identifikací odpovídajícího obličeje, získanou z názvu souboru.

## 6.9. Vyhodnocování podobnosti obličejů

Vyhodnocování podobnosti obličejů je poměrně jednoduchá matematická operace. V uvažované aplikaci má každý embedding tvar 512 rozměrného vektoru, podobnost je potom určena jako velikost vektoru, který je rozdílem dvou posuzovaných vektorů. Pomocí knihovny numpy (v kódu jako „np“) je možné získat podobnost obličejů, kterým odpovídají vektory `emb_1` a `emb_2` pomocí:

```
podobnost = np.sqrt(np.sum(np.square(np.subtract(emb_1, emb_2))))
```

Jelikož je podobnost dána velikostí rozdílového vektoru, menší hodnota značí vyšší podobnost. Empiricky bylo určeno, že hodnota 1.05 je ve většině případů vhodná hranice pro správné určení shody, obličeje s podobností menší než 1.05 tedy aplikace považuje za shodné. V případě, že aplikace po porovnání posuzovaného obličeje se seznamem známých obličejů nenalezne žádný s podobností menší než 1.05, označí tento obličej za neznámý. Ne vždy je však hodnota 1.05 vhodná, zvláště při vyšším šumu ve snímaném obraze z důvodu horšího osvětlení je dosaženo lepších výsledků při zvýšení této hranice.



Obrázek 6.4: Ukázka rozpoznání obličejů v reálném čase



(a) Me.jpg

(b) Joe.jpg [48]

(c) Hunter.jpg [48]

Obrázek 6.5: Externě dodané obrázky

## 6.10. Hodnocení aplikace

Zde popsaná aplikace dokáže s rozeznávat obličeje v reálném čase, není však vhodná pro biometrickou autentizaci, jelikož nedokáže rozlišit reálný obličej od fotografie. Má své limity a slouží spíše pro demonstraci veřejně dostupných nástrojů a zdrojů v oblasti machine learningu. K nevýhodám patří menší robustnost, tj. náchylnost na změnu podmínek. Změna světelných podmínek způsobuje sníženou přesnost v rozpoznávání obličejů, toto je důsledkem kompromisu se zpracováním obrazu v reálném čase. Pro zachování dostatečné frekvence snímaného obrazu bylo nutné v některých případech použít méně robustní nebo více primitivní metody, i přes to ale dosahuje aplikace správného rozpoznání ve většině případů.

Na obrázku 6.4 je ukázka rozpoznání 3 osob, které aplikace rozpoznala na základě externě dodaných fotografií (obrázek 6.5).

## 7. Závěr

Možnosti volně dostupné umělé inteligence jsou v současnosti velice široké. Existuje velké množství open source programů pro vývoj aplikací umělé inteligence, které umožňují jednoduchou implementaci AI prvků. Kromě open source programů je také mnoho volně dostupných zdrojů, které jsou důležité pro vývoj aplikací využívajících umělou inteligenci.

V této práci bylo cílem zjistit současný stav a možnosti vyvíjení machine learning aplikací pomocí open source softwaru. Ze všech existujících open source programů pro vytváření AI aplikací bylo popsáno v této práci několik nejpoužívanějších a na některých z nich demonstrovány možnosti jejich použití při tvorbě aplikací umělé inteligence. Vysvětleny byly základní pojmy týkající se machine learningu a také byl popsán výběr zdrojů volně dostupných dat pro učení modelů umělé inteligence.

V kapitole „Ukázky vybraných nástrojů“ bylo ukázáno několik jednoduchých programů umělé inteligence. Některé z nich využívají volně dostupné modely, u dalších byly modely umělé inteligence vytvořeny. Záměrně byly vybrány různorodé ukázky pro ilustraci širokých možností uplatnění aplikací umělé inteligence.

V rámci práce byla také vytvořena rozsáhlejší závěrečná aplikace využívající umělou inteligenci pomocí nástroje Tensorflow, tato aplikace je ukázkou toho, co je se současnými volně dostupnými nástroji možné vytvořit. Cílem bylo vytvořit aplikaci, která je schopná detekovat a rozeznávat lidské obličeje v reálném čase, tento cíl byl splněn.

Dalším krokem ve vyvíjení aplikace pro rozpoznávání obličejů v reálném čase by bylo další snížení výpočtové náročnosti a zrychlení celého procesu, tak aby aplikace mohla fungovat na méně výkonných strojích, například na mobilních zařízeních. Tohoto by mohlo být dosaženo vytvořením vlastních více efektivních funkcí pro předběžné zpracování obrazu, které by byly více vhodné pro danou aplikaci.

Další kroky v rozvíjení tématu této práce by zahrnovaly podrobnější popis jednotlivých nástrojů a větší množství rozsáhlejších ukázek použití těchto programů.

# Literatura

- [1] 15 Top Open Source Artificial Intelligence Tools. Datamation [online]. 2016 [cit. 2018-05-07]. Dostupné z: <https://www.datamation.com/open-source/slideshows/15-top-open-source-artificial-intelligence-tools.html>
- [2] JIA, Yangqing, Evan SHELHAMER, Jeff DONAHUE, et al. Caffe Tutorial. Caffe [online]. Berkeley: Berkeley AI Research, 2014 [cit. 2018-02-24]. Dostupné z: <http://caffe.berkeleyvision.org/tutorial/>
- [3] Caffe. Github [online]. San Francisco: GitHub, 2014 [cit. 2018-05-07]. Dostupné z: <https://github.com/BVLC/caffe/>
- [4] JIA, Yangqing, Evan SHELHAMER, Jeff DONAHUE, et al. Caffe Tutorial. Caffe [online]. Berkeley: Berkeley AI Research, 2014 [cit. 2018-02-24]. Dostupné z: <http://caffe.berkeleyvision.org/installation.html/>
- [5] Deep Learning For Java. Deeplearning4j [online]. San Francisco: Eclipse Deeplearning4j Development Team, 2017 [cit. 2018-05-07]. Dostupné z: <https://deeplearning4j.org/>
- [6] Eclipse Deeplearning4j. GitHub [online]. San Francisco: GitHub, 2017 [cit. 2018-05-07]. Dostupné z: <https://github.com/deeplearning4j/>
- [7] Quickstart Guide. Deeplearning4j [online]. San Francisco: Eclipse Deeplearning4j Development Team, 2017 [cit. 2018-05-07]. Dostupné z: <https://deeplearning4j.org/quickstart.html>
- [8] Visualize, Monitor and Debug Network Learning. Deeplearning4j [online]. San Francisco: Eclipse Deeplearning4j Development Team, 2017 [cit. 2018-05-07]. Dostupné z: <https://deeplearning4j.org/visualization>
- [9] H2O. GitHub [online]. San Francisco: GitHub, 2011 [cit. 2018-05-07]. Dostupné z: <https://github.com/h2oai/h2o-3>
- [10] Overview. H2O [online]. Mountain View: H2O.ai, 2011 [cit. 2018-05-07]. Dostupné z: <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html>
- [11] Using Flow - H2O's Web UI. H2O [online]. Mountain View: H2O.ai, 2011 [cit. 2018-05-07]. Dostupné z: <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/flow.html>
- [12] Algorithms. H2O [online]. Mountain View: H2O.ai, 2011 [cit. 2018-05-07]. Dostupné z: <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science.html#data-science>
- [13] CNTK FAQ. Microsoft Docs [online]. Redmond: Microsoft, 2014 [cit. 2018-05-07]. Dostupné z: <https://docs.microsoft.com/en-us/cognitive-toolkit/CNTK-FAQ>
- [14] The Microsoft Cognitive Toolkit. Microsoft Docs [online]. Redmond: Microsoft, 2014 [cit. 2018-05-07]. Dostupné z: <https://docs.microsoft.com/cs-cz/cognitive-toolkit/>
- [15] CNTK. GitHub [online]. San Francisco: GitHub, 2014 [cit. 2018-05-07]. Dostupné z: <https://github.com/Microsoft/CNTK/>

## LITERATURA

- [16] Theano at a Glance. Deep Learning [online]. Montreal: Montreal Institute for Learning Algorithms, 2009 [cit. 2018-05-07]. Dostupné z: <http://deeplearning.net/software/theano/introduction.html>
- [17] Windows Installation Instructions. Deep Learning [online]. Montreal: Montreal Institute for Learning Algorithms, 2009 [cit. 2018-05-07]. Dostupné z: [http://deeplearning.net/software/theano/install\\_windows.html](http://deeplearning.net/software/theano/install_windows.html)
- [18] Theano. GitHub [online]. San Francisco: GitHub, 2009 [cit. 2018-05-07]. Dostupné z: <https://github.com/Theano/Theano>
- [19] Introducing Apache Mahout. IBM [online]. Armonk: IBM, 2009 [cit. 2018-05-07]. Dostupné z: <https://www.ibm.com/developerworks/java/library/j-mahout/>
- [20] Mahout. GitHub [online]. San Francisco: GitHub, 2008 [cit. 2018-05-07]. Dostupné z: <https://github.com/apache/mahout>
- [21] Mahout. Apache Mahout [online]. Forest Hill: Apache Software Foundation, 2008 [cit. 2018-05-07]. Dostupné z: <https://mahout.apache.org/docs/latest/index.html>
- [22] MLlib. Apache Spark [online]. Forest Hill: Apache Software Foundation, 2016 [cit. 2018-05-07]. Dostupné z: <https://spark.apache.org/mllib/>
- [23] Introduction. DLib C++ Library [online]. Boston: Davis E. King, 2002 [cit. 2018-05-07]. Dostupné z: <http://dlib.net/intro.html>
- [24] Machine Learning. DLib C++ Library [online]. Boston: Davis E. King, 2002 [cit. 2018-05-07]. Dostupné z: <http://dlib.net/ml.html>
- [25] DLib. GitHub [online]. San Francisco: GitHub, 2008 [cit. 2018-05-07]. Dostupné z: <https://github.com/davisking/dlib>
- [26] DLib models. GitHub [online]. San Francisco: GitHub, 2008 [cit. 2018-05-07]. Dostupné z: <https://github.com/davisking/dlib-models/commits?author=davisking>
- [27] KING, Davis. Dlib-ml: A Machine Learning Toolkit. In: The Journal of Machine Learning Research [online]. Boston: JMLR, 2009 [cit. 2018-05-07]. ISSN 1533-7928. Dostupné z: <http://jmlr.csail.mit.edu>
- [28] Implementations. Numenta [online]. Redwood City: Numenta, 2013 [cit. 2018-05-07]. Dostupné z: <https://www.numenta.org/implementations/>
- [29] Network API. NuPIC Docs [online]. Redwood City: Numenta, 2013 [cit. 2018-05-07]. Dostupné z: <http://nupic.docs.numenta.org/1.0.3/api/network/index.html>
- [30] HAWKINS, Jeff. HIERARCHICAL TEMPORAL MEMORY [online]. In: . Redwood City: Numenta, 2011 [cit. 2018-05-07]. Dostupné z: <https://numenta.com/assets/pdf/whitepapers/hierarchical-temporal-memory-cortical-learning-algorithm-0.2.1-en.pdf>
- [31] NuPIC.core. GitHub [online]. San Francisco: GitHub, 2014 [cit. 2018-05-07]. Dostupné z: <https://github.com/numenta/nupic.core>

- [32] TensorFlow. TensorFlow [online]. Mountain View: Google Brain Team, 2017 [cit. 2018-05-07]. Dostupné z: <https://www.tensorflow.org/>
- [33] API Documentation. TensorFlow [online]. Mountain View: Google Brain Team, 2017 [cit. 2018-05-07]. Dostupné z: [https://www.tensorflow.org/api\\_docs/](https://www.tensorflow.org/api_docs/)
- [34] TensorFlow. GitHub [online]. San Francisco: GitHub, 2017 [cit. 2018-05-07]. Dostupné z: <https://github.com/tensorflow/tensorflow>
- [35] TensorBoard. GitHub [online]. San Francisco: GitHub, 2017 [cit. 2018-05-07]. Dostupné z: <https://github.com/tensorflow/tensorboard>
- [36] COLLOBERT, Ronan, Clement FARABET, Koray KAVUKCUOGLU a Soumith CHINTALA. Torch. Torch [online]. 2002 [cit. 2018-05-07]. Dostupné z: <http://torch.ch/>
- [37] Torch. GitHub [online]. San Francisco: GitHub, 2008 [cit. 2018-05-07]. Dostupné z: <https://github.com/torch/torch7/wiki/Cheatsheet>
- [38] Windows. GitHub [online]. San Francisco: GitHub, 2008 [cit. 2018-05-07]. Dostupné z: <https://github.com/torch/torch7/wiki/Windows>
- [39] Lua. Lua [online]. Rio de Janeiro: Departamento de Informática, PUC-Rio, 1993 [cit. 2018-05-07]. Dostupné z: <https://www.lua.org/about.html>
- [40] Kaggle [online]. Kaggle, 2010 [cit. 2018-05-07]. Dostupné z: <https://www.kaggle.com>
- [41] UCI Machine Learning Repository. UCI Machine Learning Repository [online]. Irvine: University of California, 1987 [cit. 2018-05-07]. Dostupné z: <https://archive.ics.uci.edu/ml/index.php>
- [42] Mldata.org [online]. Berlin: TU Berlin, 2009 [cit. 2018-05-07]. Dostupné z: <http://ml-data.org/repository/data/>
- [43] Registry.opendata.aws/ [online]. Seattle: Amazon Web Services, 2017 [cit. 2018-05-07]. Dostupné z: <https://registry.opendata.aws/>
- [44] Public data sets for testing and prototyping. Microsoft Azure [online]. Redmond: Microsoft Azure, 2017 [cit. 2018-05-07]. Dostupné z: <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-public-data-sets>
- [45] Google BigQuery Public Datasets. Google Cloud [online]. Mountain View: Google, 2017 [cit. 2018-05-07]. Dostupné z: <https://cloud.google.com/bigquery/public-data/>
- [46] Quandl.com [online]. Toronto: Quandl, 2013 [cit. 2018-05-07]. Dostupné z: <https://www.quandl.com/>
- [47] Datacatalog.worldbank.org [online]. Washington: The World Bank [cit. 2018-05-07]. Dostupné z: <https://datacatalog.worldbank.org/>
- [48] img.washingtonpost.com [online]. Worldviews [cit. 2018-05-07]. Dostupné z: <https://img.washingtonpost.com/blogs/worldviews/files/2014/05/AP090106023915.jpg>

## LITERATURA

- [49] ILSVRC2014. Image Net [online]. Image Net, 2014 [cit. 2018-05-07]. Dostupné z: <http://www.image-net.org/challenges/LSVRC/2014/results>
- [50] Googlenet in Keras [online]. San Francisco: github.io, 2016 [cit. 2018-05-07]. Dostupné z: [http://joelouismarino.github.io/blog\\_posts/blog\\_googlenet\\_keras.html](http://joelouismarino.github.io/blog_posts/blog_googlenet_keras.html)
- [51] JAMES, G., D. WITTEN, T. HASTIE a R. TIBSHIRANI. An Introduction to Statistical Learning. New York: Springer-Verlag New York, 2013. ISBN 978-1-4614-7138-7.
- [52] BARBER, D. Bayesian Reasoning and Machine Learning. 1. Cambridge: Cambridge University Press, 2012. ISBN 113911655X.
- [53] HASTIE, Trevor, Robert TIBSHIRANI a Jerome FRIEDMAN. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics). 2nd edition. Springer, 2016. ISBN 0387848576.
- [54] VAN VEEN, Fjodor. The Neural Network Zoo. The Asimov Institute [online]. Utrecht: The Asimov Institute, 2016 [cit. 2018-05-07]. Dostupné z: <http://www.asimovinstitute.org/neural-network-zoo/>
- [55] VAN GERVEN, Marcel a Sander BOHTE. Artificial Neural Networks as Models of Neural Information Processing. Frontiers Media, 2017. ISBN 9782889454013.
- [56] SCHROFF, Florian, Dmitry KALENICHENKO a James PHILBIN. FaceNet: A Unified Embedding for Face Recognition and Clustering [online]. 2015 [cit. 2018-05-07]. Dostupné z: <https://arxiv.org/abs/1503.03832>. Cornell University.
- [57] Face Recognition using Tensorflow. GitHub [online]. San Francisco: GitHub, 2017 [cit. 2018-05-07]. Dostupné z: <https://github.com/davidsandberg/facenet>
- [58] CAO, Qiong, Li SHEN, Weidi XIE, Omkar PARKHI a Andrew ZISSERMAN. VGG-Face2: A dataset for recognising faces across pose and age [online]. Oxford, 2017 [cit. 2018-05-07]. Dostupné z: <https://arxiv.org/abs/1710.08092>. University of Oxford.
- [59] LECUN, Yann, Corinna CORTES a Christopher BURGES. THE MNIST DATABASE [online]. New York, 1998 [cit. 2018-05-07]. Dostupné z: <http://yann.lecun.com/exdb/mnist/>
- [60] Tensorflow models [online]. San Francisco: GitHub, 2016 [cit. 2018-05-07]. Dostupné z: <https://github.com/tensorflow/models/tree/master/tutorials/image/imagenet>

# Seznam obrázků

2.1	Feed forward neuronové sítě [54]	4
2.2	Konvoluční neuronová síť [54]	4
2.3	Rekurentní neuronová síť [54]	5
3.1	Caffe logo [2]	6
3.2	Deeplearning4j logo [5]	7
3.3	H2O logo [9]	8
3.4	Theano logo [16]	9
3.5	DLib logo [23]	11
3.6	Tensorflow logo [32]	12
4.1	Kaggle logo [40]	14
5.1	Rozpoznání předmětů pomocí Tensorflow aplikace	17
5.2	Stabilizace obličeje pomocí knihovny DLib	18
5.3	Závislost přesnosti klasifikace na počtu cyklů učení	19
5.4	Klasifikované obrázky číslic	19
6.1	Detekce obličejových prvků pomocí DLib [48]	22
6.2	Schéma neuronové sítě Inception ResNet v1 [50]	23
6.3	Schéma algoritmu učení [56]	23
6.4	Ukázka rozpoznání obličejů v reálném čase	25
6.5	Externě dodané obrázky	25

# Seznam zkratek a symbolů

AI	Artificial Intelligence
API	Application programming interface
np	knihovna numpy
CPU	procesor
GPU	grafický procesor
JVM	Java Virtual Machine
JDK	Java Development Kit

# Seznam příloh na CD

## Příloha 1 - složka Caffe ukázka

train.prototxt	neuronová síť pro učení
solver.prototxt	řešič
deploy.prototxt	neuronová síť pro předpověď
Caffemodel	složka obsahující model

## Příloha 2 - složka Tensorflow ukázka

video.py	zdrojový kód programu
classify_image.py	soubor s funkcí předpovědi
imagenet	složka obsahující model

## Příloha 3 - složka DLib ukázka

DLib_face_alignment.py	zdrojový kód programu
shape_predictor_68_face_landmarks.dat	soubor nástroje DLib pro detekci obličejových prvků
img.jpg	obrázek ke zpracování

## Příloha 4 - složka Theano ukázka

theano2.py	zdrojový kód programu
MNIST	složka obsahující soubory datasetu MNIST
img	složka obsahující ukázkové obrázky

## Příloha 5 - složka Aplikace

real_time2.py	zdrojový kód programu
face_prepro.py	zdrojový kód obsahující funkce pro předběžné zpracování obrazu
model	složka obsahující model
haarcascade_frontalface_default.XML	soubor nástroje OpenCV pro detekci obličeje
shape_predictor_68_face_landmarks.dat	soubor nástroje DLib pro detekci obličejových prvků
Faces	složka pro externě dodané obrázky