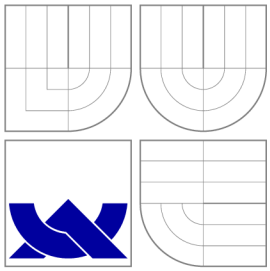
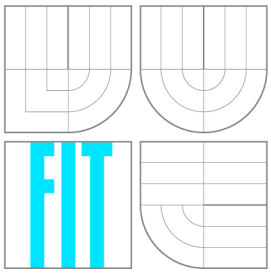


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A ROZPOZNÁNÍ OBJEKTŮ V OBRAZE

OBJECT DETECTION AND RECOGNITION IN IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAELA MUZIKÁŘOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Dr. Ing. PAVEL ZEMČÍK

BRNO 2015

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací aplikace typu klient-server, která umožňuje rozpoznání objektů v obraze a využívá již existující mobilní aplikaci. V teoretické části jsou nejprve popsány rozdíly lidského a počítačového vidění, dále detekce a rozpoznání objektů včetně vybraných metod. Další sekce obsahuje popis umělých neuronových sítí, které byly pro práci hlouběji nastudovány, spolu s jejich využitím k rozpoznání objektů. Následují informace, týkající se vybraných mobilních aplikací pro rozpoznání objektů v obraze, zakončené přehledem frameworků a knihoven, umožňujících práci s neuronovými sítěmi. Z nich byl k práci zvolen Caffe Framework. Dále je popsán průběh návrhu a řešení a vytvořený systém včetně experimentů a datasetů, použitých k ověření jeho funkčnosti.

Abstract

This bachelor's thesis deals with design and implementation of client-server application for object recognition with the use of existing mobile application. Theoretical part describes the differences between human and computer vision, followed by information about object detection and recognition with selected methods. The next section provides a detailed overview of artificial neural networks, which were used for this work, with their qualities for object recognition. Following part examines selected mobile applications for object recognition, followed by existing frameworks and libraries with focus on artificial neural networks. Among these, Caffe Framework was selected for the work. The next section illustrates the progress of design and implementation and describes the system, along with experiments and dataset used to prove its functionality.

Klíčová slova

počítačové vidění, detekce objektů, rozpoznání objektů, klasifikace, Caffe framework, umělé neuronové sítě, konvoluční neuronové sítě

Keywords

computer vision, object detection, object recognition, classification, Caffe framework, artificial neural networks, convolutional neural networks

Citace

Michaela Muzikářová: Detekce a rozpoznání objektů v obraze, bakalářská práce, Brno, FIT VUT v Brně, 2015

Detekce a rozpoznání objektů v obraze

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Prof. Dr. Ing. Pavla Zemčíka a uvedla jsem všechny literární prameny, ze kterých jsem čerpala.

.....
Michaela Muzikářová
17. května 2015

Poděkování

Mé poděkování patří panu Prof. Dr. Ing. Pavlu Zemčíkovi za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval.

Obsah

Kapitola 1

Úvod

Počítače, potažmo fotoaparáty pro nás v dnešní době hrají zásadní roli a málokdo by si dokázal představit svůj život bez nich. Jejich sílu a schopnosti můžeme v mnoha případech využít v náš prospěch. Denně přicházíme do styku s okolním světem, ve kterém jsme stavěni před úkol rozpoznávání objektů – i s tímto úkolem nám mohou počítače pomoci. Člověk dokáže naprosto přirozeně poznávat různé známé i neznámé objekty bez ohledu na jejich velikost, umístění v zorném poli nebo úhel pohledu, a poznává také objekty, které do zorného pole nezasahují celé. Pro počítač ale není podobný úkol vůbec samozřejmý, a právě proto se touto problematikou zabývá obor počítačového vidění.

Počítačové vidění je v dnešní době jednou z velmi zajímavých a rychle se rozvíjejících oblastí lidského zkoumání. Díky rostoucím výpočetním schopnostem je možné různé úlohy počítačového vidění provádět čím dál rychleji a efektivněji, což rozšiřuje možnosti jeho dalšího využití. S pomocí strojového učení a počítačového vidění je již dnes možné například vytvářet autonomní vozidla, schopná provozu bez nehod, ovládat roboty, prozkoumávající cizí planety, analyzovat medicínská data do největších detailů a vykonávat mnohé další činnosti, o kterých se lidem ještě před několika desetiletími ani nesnilo. Síly těchto oborů lze využít nejen v mnoha různých vědeckých odvětvích, ale i pro zábavu (ovládání her pohybem). Hranice jejich využití jsou určené pouze výpočetními kapacitami moderních počítačů, našimi schopnostmi a fantazií, takže se s každým dnem a s každým dalším výzkumem posouvají dále.

Počítačové vidění i strojové učení mohou také sloužit pro praktické úkoly, jakými jsou kontrola procesů, automatizace výroby nebo různé typy rozpoznávání (kouřové čidlo, kamerový systém). Jejich přístupy mohou lidem pomáhat s bezpečností (detekce a rozpoznání dopravních značek v kameře auta), s ostřením u fotoaparátu (detekce obličejů), výzkumem (rozpoznání druhů rostlin, tomografické snímky) nebo mohou asistovat například zrakově postiženým osobám při orientaci ve světě, která by pro ně jinak byla velmi obtížná. K tomuto účelu slouží různé aplikace pro rozpoznávání předmětů, textu nebo pro hlasové ovládání počítače.

Obor rozpoznávání objektů v obraze mě zaujal, protože je fascinující, k čemu všemu je možné tuto oblast využít. Rozpoznávat lze téměř cokoliv od poznávacích značek aut přes významné budovy ve městě až po druhy ovoce. Poslední jmenované může být dobře využitelné v podobě nákupní aplikace pro zrakově postižené osoby a může jim tak pomoci při nákupu.

Návrh a implementace podobné aplikace, sloužící k rozpoznávání ovoce, je cílem této bakalářské práce. Hlavní snahou při vývoji byla jednoduchá obsluha aplikace ze strany uživatele, efektivní komunikace jednotlivých částí aplikace a co nejlepší úspěšnost aplikace

při rozpoznávání vybraných druhů ovoce.

Práce je členěna následovně. V následující kapitole jsou shrnuty nejdůležitější informace, týkající se aktuální situace v oblastech detekce a rozpoznávání objektů v obraze včetně srovnání lidského a počítačového vidění. V kapitole 3 jsou popsány nejdůležitější pojmy, vztahující se k oblasti rozpoznávání objektů v obraze a také k umělým neuronovým sítím s možnostmi jejich využití v této oblasti. Kapitola 4 popisuje existující knihovny a frameworky, umožňující efektivní tvorbu umělých neuronových sítí, a dále pojednává o existujících mobilních klient-server aplikacích se zaměřením na rozpoznávání objektů v obraze. V kapitole 5 je popsán návrh a implementace výsledné aplikace a také použitá sada obrazových dat, se kterou aplikace pracuje. Kapitola 6 obsahuje popis experimentů, provedených s aplikací. Poslední část práce zahrnuje diskuzi dosažených výsledků a možnosti dalšího pokračování práce.

Kapitola 2

Počítačové a lidské vidění

V této kapitole jsou popsány základní informace, týkající se počítačového vidění, strojového učení, a dále lidského a počítačového vnímání světa. Poslední část kapitoly obsahuje základní informace o klasifikaci. Nejedná se o úplný přehled, ale pouze o shrnutí klíčových informací, důležitých pro tuto bakalářskou práci.

Detekce a rozpoznávání objektů v obraze je v současné době na takové úrovni, jež umožňuje jejich praktické využití v mnoha oblastech lidské činnosti. V state-of-the-art přístupech se dnes využívá mnoho různých metod detekce a rozpoznávání objektů v obraze, mezi něž patří také umělé neuronové sítě, popsané v kapitole 3. Sítě již prošly velkým vývojem a jsou velmi dobře použitelné pro rozpoznávání objektů v obraze také díky algoritmu zpětné propagace, který vznikl v 80. letech minulého století. Tyto sítě jsou hlavním předmětem práce.

Ačkoliv je na trhu dnes velké množství mobilních klient-server aplikací s různými zaměřeními v oblasti detekce a rozpoznávání objektů v obraze, chtěla jsem si sama vyzkoušet návrh a implementaci podobné aplikace s tím, že by později mohla sloužit zrakově postiženým lidem při výběru ovoce nebo i zeleniny v obchodě. Několik existujících aplikací pro rozpoznávání objektů v obraze je popsáno v kapitole 4, kde je uvedena i podobně zaměřená aplikace, pomáhající slepým lidem s rozpoznáváním objektů denní potřeby.

2.1 Počítačové vidění a strojové učení

Počítačové vidění je obor, spadající pod umělou inteligenci. Jedná se o oblast, snažící se popsat svět, zachycený na fotografiích nebo videích, a rekonstruovat ve 2D i 3D jeho vlastnosti (například tvary předmětů, rozložení barev či osvětlení). Součástí tohoto procesu může být i detekce objektů v obraze a jejich rozpoznávání. Díky porozumění vlastnostem reálného světa, obrazu a senzoru (např. kamera) je možné z obrazu vyvozovat užitečné informace, jakými jsou například velikost překážky před mobilním robotem na Marsu, identita osoby, kontrolované hlídacím systémem [18] nebo poloha nádoru při snímání těla magnetickou rezonancí [15]. Počítačové vidění má dnes velmi široké využití, jako je například 3D modelování [13], zobrazování lékařských dat [2], biometrie [11], rozpoznávání gest [49] či řízení dopravy v reálném čase [12].

S oborem počítačového vidění úzce souvisí i další podoblast umělé inteligence – strojové učení. Tato oblast je někdy označována také jako statistické učení nebo rozpoznávání vzorů. Zabývá se algoritmy, které se dokáží učit z předložených dat, přičemž učení v tomto kontextu znamená, že se vytvořené programy chovají určitým požadovaným způsobem, aniž by

k němu byly explicitně naprogramovány. Strojové učení souvisí také se statistikou a dobýváním znalostí a jeho metody se používají například v rozpoznávání řeči a psaného textu či v biomedicínské informatice (systémy pro podporu rozhodování), robotice a mnoha dalších oblastech [33].

2.2 Fáze lidského zrakového vnímání a počítačového vidění

I když je pro člověka vnímání okolního světa subjektivně naprosto přirozenou záležitostí, pro mozek to není tak jednoduché a proto šedá kůra mozková, zpracovávající zrakové vjemy, zabírá přibližně 2/3 mozku. Zrakové vnímání je důležité pro získávání informací o okolním prostředí například za účelem nalezení potravy, vyhnutí se překážce či identifikaci hrozícího nebezpečí [1].

Lidské zrakové vnímání lze rozdělit do fází detekce, diskriminace, a identifikace objektů. Detekce objektu označuje určení jeho výskytu v zorném poli. Diskriminace je pak rozlišením mezi větším počtem objektů, jako například při hledání mince v peněžence. Identifikace objektu značí určení identity objektu, například rozpoznání člověka na fotografii. Tyto tři roviny se však v praxi často prolínají.

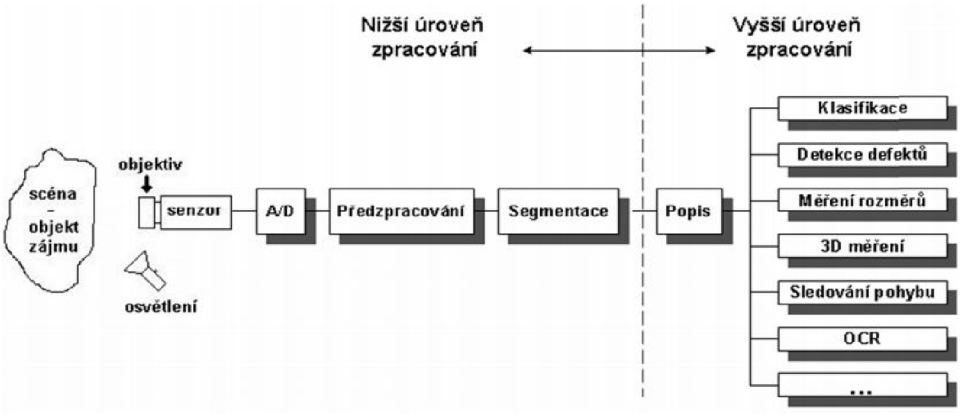
Při detekci, diskriminaci a identifikaci objektů je stále člověk daleko před počítači, ačkoliv počítače mají mnohem vyšší procesní kapacity. Je to z důvodu potenciálních nástrah, které vidění skýtá, k nimž patří následující:

- Převod z 3D do 2D nevyhnutelně přináší ztrátu informace kvůli vlastnostem perspektivní transformace.
- Je nutné zahrnout interpretaci.
- V každém obraze je obsažen pouze lokální pohled; je však nutné pohlížet na věci také globálně, přičemž lidský mozek je schopný si doplnit a domyslet podobu objektů, které i jen z části zasahují do zorného pole.

K úspěšnému rozpoznání objektu je dále nutné disponovat v mysli dostatečně širokým paměťovým asociačním skladem, který obsahuje možné tvary, velikosti a další vlastnosti každého známého objektu. S tímto skladem jsou objekty v zorném poli porovnávány. Člověk často objekty rozpoznává právě podle tvaru, který je specifikuje. V některých případech se ale využívají i jiné indicie, jako je například vzor srsti nebo barva kůže u zvířat. Určující může být i poloha objektu vzhledem k celkovému kontextu (např. umístění kliky na dveřích) nebo typický pohyb objektu (např. motýl v letu) [50].

Počítačové vidění se pokouší porozumět scéně, kterou lidské oko vnímá naprosto samozřejmě, ať už se jedná o výše zmíněnou detekci, diskriminaci či identifikaci objektů. Je také velmi silně propojeno s umělou inteligencí, počítačovými vědami, zpracováním signálů a rozpoznáváním obrazu. Na rozdíl od počítačové grafiky, kde se vytváří obraz skládáním jednoduchých tvarů (například úsečky nebo kružnice) a obrazová data jsou bez šumu, zpracovává počítačové vidění obraz skutečného světa včetně šumu a dalších možných zkreslení.

Proces zpracování obrazu, popsaný na obrázku 2.1, lze rozdělit do dvou základních úrovní – nižší a vyšší. Nižší úroveň zahrnuje snímání, digitalizaci, předzpracování a segmentaci obrazu. Mezi fáze vyšší úrovně patří popis obrazu a následně podle účelu aplikace například detekce objektů, jejich klasifikace, rozpoznávání znaků, sledování pohybu a podobně [23]. Následuje podrobnější popis jednotlivých fází procesu.



Obrázek 2.1: Fáze procesu zpracování obrazu [23].

1. Snímání a digitalizace zahrnuje uložení obrazu v číselné formě do počítače. Při snímání obrazu je klíčové osvětlení scény, fotoaparát nebo kamera a objektiv, kterým je obraz snímán. Pro digitalizaci je nejdůležitějším prvkem fotoaparátu či kamery senzor, přičemž na jeho kvalitě do značné míry závisí konečný výsledek (například množství šumu v obraze).

2. Předzpracování obrazu sestává z potlačení šumu, zkreslení a jiných nežádoucích rysů obrazu, které vznikají při jeho přenosu. Pro určité algoritmy či postupy může předzpracování zahrnovat například zvýraznění hran v obraze pro další zpracování.

3. Segmentace značí rozdělení obrazu na více částí, které jej pokrývají, s cílem nalézt v obraze objekty, zajímavé pro další zpracování. Při segmentaci se využívá znalosti interpretace obrazu. Vstupem tohoto procesu je obraz, výstup se liší podle použité segmentační metody a mohou jím být například části obrazu nebo poloha určitého objektu v obraze. Mezi metody segmentace patří například segmentace na základě barev, transformační metody, prahování nebo texturové filtry.

4. Následuje popis objektů, nalezených v obraze, který může být kvalitativní (relace mezi objekty – například seznam primitiv obrazu) či kvantitativní (číselné charakteristiky – například vektor příznaků).

5. Poslední částí procesu zpracování obrazu je fáze nejvyšší úrovně, vedoucí k porozumění jeho obsahu. Konkrétní volba závisí na účelu aplikace. Může se jednat například o klasifikaci objektů do tříd podle tvarů (kulaté, hranaté, apod.), měření rozměrů objektů nebo sledování pohybu [16].

Kapitola 3

Klasifikace a neuronové sítě pro rozpoznávání objektů v obraze

V této kapitole se nachází informace o detekci a rozpoznání objektů a jejich klasifikaci. Následují údaje o metodě učení Boosting, původně uvažované pro tuto práci. Následující části popisují hlavní prostředky, zásadní pro tuto práci – neuronové sítě. Sekce 3.8 popisuje konvoluční neuronové sítě, které byly přímo použity pro práci. Kapitola neposkytuje úplný přehled, ale nachází se v ní pouze informace relevantní pro práci.

3.1 Detekce, rozpoznání objektů a klasifikace

Detekce objektů se zabývá nalezením objektů v obraze, rozpoznávání objektů pak určením jejich identity za pomocí množiny známých označení (angl. *label*). Objekty patří do některé z tříd, kterými mohou být například lidé, auta, budovy; v případě této bakalářské práce jsou třídami druhy ovoce [48].

Přístupů k detekci objektů existuje velké množství. Jedna z nejpopulárnějších metod byla vytvořena Violou a Jonesem a využívá Haarovy příznaky (založené na histogramu orientovaných gradientů) a mechanismus posuvného okna [46]. Tato metoda je implementována v OpenCV [36] knihovně, popsané v kapitole 4. Ačkoliv je její původní využití rozpoznávání tváří, je možné s její pomocí rozpoznávat i jiné objekty. Dalším přístupem je detekce založená na podobnosti, při které jsou objekty zájmu rozděleny do množin pozitivních a negativních příkladů a rozhodování o tom, zda je neznámý objekt objektem zájmu dělá klasifikátor (funkční blok, založený na strojovém učení), který je postupně aplikován na podokna obrazu [22]. Poslední jmenovaný způsob detekce využívá například algoritmus AdaBoost [14], popsaný v sekci 3.2. Další možností je využití neuronových sítí, které jsou předmětem této práce.

Klasifikace je jedním z úkolů dolování dat. Jejím cílem je zařadit nové pozorování do některé z klasifikačních tříd. Toto zařazení probíhá na základě trénovací množiny, obsahující pozorování, jejichž příslušnost k některé ze tříd je již známa. Klasifikátor je pak učící algoritmus, implementující klasifikaci daných dat.

Klasifikace může být binární nebo do více tříd. Binární klasifikace je nejjednodušší případ – může se například jednat o rozhodnutí, zda je doručený e-mail spam nebo ne. U klasifikace do více tříd může objekt náležet do jedné nebo i do více z nich. Třídami mohou být například krevní skupiny nebo druhy ovoce. Klasifikace využívající více tříd často používá kombinaci několika binárních klasifikátorů.

Mezi základní techniky klasifikace patří například metody pravděpodobnostní, lineární, rozhodovací strom, genetické algoritmy či metody založené na fuzzy logice.

V praxi existují 2 základní přístupy ke klasifikaci dat:

- Učení s učitelem (angl. *supervised learning*, označováno také jako klasifikace nebo shluková analýza), při němž jsou k dispozici třídy a cílem je vytvořit pravidlo, které umožní klasifikovat nové pozorování do jedné z nich. Data jsou zde rozdělena do 2 nebo více skupin, mezi něž patří také trénovací a testovací data. Jako trénovací data jsou učícímu algoritmu předkládány příklady a protipříklady objektů, které se má naučit rozpoznávat. Pomocí testovacích dat se po fázi tréninku ověřuje úspěšnost algoritmu.

- Učení bez učitele (angl. *unsupervised learning*, označováno také jako shlukování), při němž je k dispozici sada pozorování a cílem je vytvořit z těchto dat třídy a tím je popsat [34].

Klasifikátory lze v oblasti rozpoznání objektů v obraze využít k rozhodování, zda část obrazu obsahuje objekt zájmu. Typicky jsou trénovány k detekci objektů, které jsou ve středu klasifikovaného podobrazu, mají normalizovanou velikost a správné zarovnání. Detekce objektů se proto provádí buďto pomocí klasifikace obsahu všech podoken, která by mohla obsahovat objekt zájmu, nebo klasifikací všech možných podoken obrazu. Toto se obvykle realizuje „skenováním“ obrazu pomocí posuvného okna pevných rozměrů, kde se pro každou polohu okna provádí klasifikace jeho obsahu; pokud je nalezen objekt zájmu, je tato pozice okna považována za výstup detekce.

Pro snížení výpočetní náročnosti je někdy možné předzpracovat analyzovaný obraz. Tento proces slouží k identifikaci těch částí obrazu, kde se objekty zájmu nemohou nacházet – tyto části lze vyloučit z klasifikačního procesu. Příkladem předzpracování může být květina, která nebude v části obrazu, jež obsahuje nebe; lidská tvář zase nebude v části obrazu, jež neobsahuje barvu kůže.

Tato metoda detekce je však závislá na orientaci a velikosti objektu, ale často je potřebné detekovat objekty bez ohledu na tyto jejich vlastnosti. Někdy je proto možným přístupem opakovaný detekční proces pro různá měřítka nebo orientace objektů. Hlavním důvodem je, že obecně vzato slabé klasifikátory nejsou invariantní vůči rotaci, měřítku či posunu. Proto by měl být detekční proces aplikován opakovaně, kvůli pokrytí různých možných rotací, měřítek a dalších vlastností objektů. Hustota vzorkování obrazu závisí na toleranci klasifikátoru vzhledem k rotaci, měřítku a dalším [22].

3.2 Boosting algoritmy Adaboost a Waldboost

Boosting je název třídy metod učení, které využívají postupnou aplikací slabých klasifikátorů (klasifikátorů s malou úspěšností klasifikace) na opakovaně modifikované verze dat. Predikce těchto slabých klasifikátorů se poté zkombinují pro vytvoření silného klasifikátoru.

Boosting se pokouší zvyšovat (angl. *boost*) přesnost zadaného učicího algoritmu. Původně byl vytvořen pro klasifikační problémy, ale lze jej využít i pro regresi [38].

AdaBoost (zkratka z *Adaptive Boosting*) je algoritmus (využívající techniku Boosting), založený na vytvoření lineární kombinace více slabých klasifikátorů do jednoho silného klasifikátoru, který je úspěšnější než použité slabé klasifikátory. Slabý klasifikátor je takový klasifikátor, jehož úspěšnost je jen o něco málo lepší než při náhodném hádání (nad 50%). AdaBoost pracuje metodou více iterací, při kterých se přidávají postupně slabé klasifikátory,

jejichž pomocí je vytvořen složený silný klasifikátor. Takovýto silný klasifikátor má vyšší přesnost než všechny použité slabé klasifikátory. Adaboost má mnoho různých modifikací.

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{3.1}$$

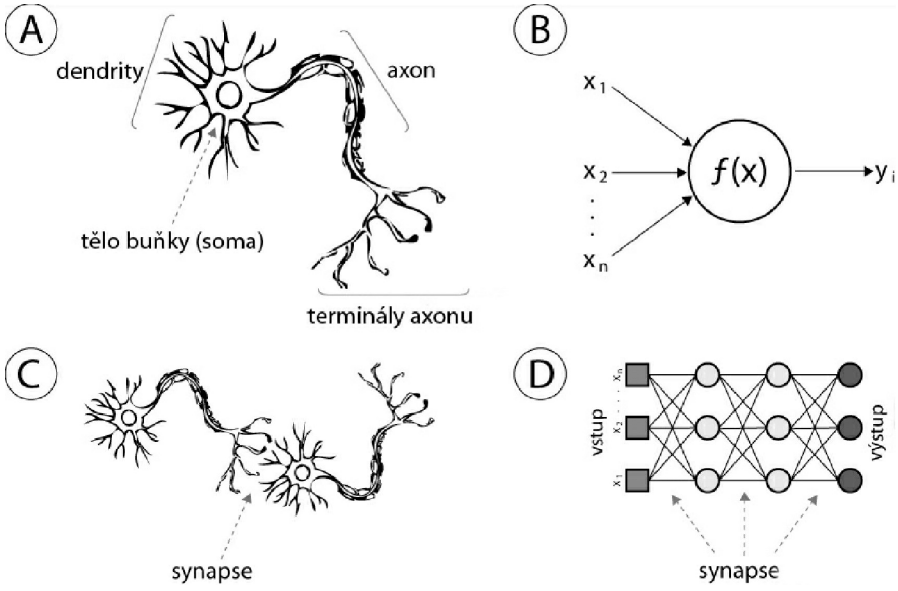
Rovnice 3.1 popisuje průběh algoritmu AdaBoost, jehož výsledkem je rozhodnutí mezi 2 alternativami (např. -1 a 1).

Mezi výhody algoritmu AdaBoost patří jeho jednoduchá implementace. Algoritmus je navíc možné zkombinovat s jinými metodami klasifikace, jako jsou například neuronové sítě, rozhodovací stromy nebo klasifikace podle nejbližších sousedů (angl. *nearest-neighbor*. Pokud jsou ale slabé klasifikátory relativně silné (chyba se zmenšuje velmi rychle), AdaBoost může být nepoužitelný [14].

WaldBoost je pravděpodobně nejdůležitější modifikací algoritmu AdaBoost. Je založený na Waldově algoritmu postupného rozhodování v kombinaci s AdaBoostem. Hlavní výhodou WaldBoostu je výrazný nárůst výkonu v porovnání s klasifikátory, založenými na AdaBoostu, prakticky beze změny v kvalitě klasifikace. WaldBoost byl poprvé použit na problém detekce obličejů [22].

3.3 Umělé neuronové sítě a jejich základní model

Neuron je specializovaná buňka, tvořící základní stavební prvek lidského nervového systému. Úkolem neuronu je přenos, zpracování a uchování informací. Skládá se z těla (*soma*), krátkých výběžků (*dendrit*) a jednoho dlouhého výběžku (*axon*). Axon slouží pro přenos vzruchů směrem ven z neuronu, dendrity dovnitř [47]. Pro přenos informací slouží synapse, které tvoří spojení mezi neurony.

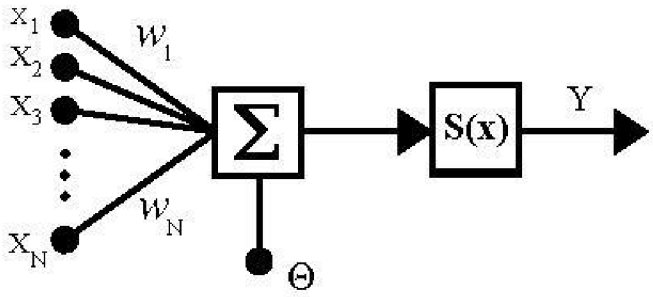


Obrázek 3.1: Neuron a synapse. Přejato a upraveno z [30].

Termín umělé neuronové sítě označuje matematické modely, založené na výše popsaných

neuronech v lidském mozku [25]. Dále budou v práci označovány pouze jako neuronové sítě. Základy sítím položili již ve 40. letech 20. století McCulloch a Pitts, kteří ve své práci popsali jeden z nejpoužívanějších modelů formálního neuronu, popsaný níže. Neuronové sítě jsou určené pro řešení problémů například z oblastí počítačového vidění nebo rozpoznávání řeči, které jsou obtížně řešitelné pomocí klasického programování. Informace je v nich typicky zpracovávána paralelně v různých uzlech (neuronech) podobně jako v jejich biologickém vzoru, přičemž synapse jsou zde reprezentovány propojeními mezi jednotlivými neurony. Klíčovými vlastnostmi neuronových sítí jsou schopnost učení, adaptace a provádění změn ve vlastní struktuře – díky tomu jsou sítě velmi užitečným modelem umělé inteligence. Navíc se jedná o adaptivní systém, jenž dokáže měnit svou vnitřní strukturu na základě procházejících informací. Změny se dosahuje pomocí úpravy vah, příslušejících spojením mezi jednotlivými neurony [41].

Na obrázku 3.1 je v části A zobrazen neuron tak, jak vypadá v lidském mozku. Část B ukazuje základní model formálního neuronu. Části C a D po řadě zobrazují synapse v lidském mozku a v umělé neuronové síti.



Obrázek 3.2: Schéma formálního neuronu dle Mccullocha a Pittse [32]

Neuronová síť se skládá z tzv. formálních neuronů. Formální neuron používá matematické funkce namísto biologických. Jedním z nejpoužívanějších modelů formálního neuronu je McCulloch-Pitts model (zkratka MCP), který je na obrázku 3.2. Podle tohoto modelu lze neuron popsat vztahem:

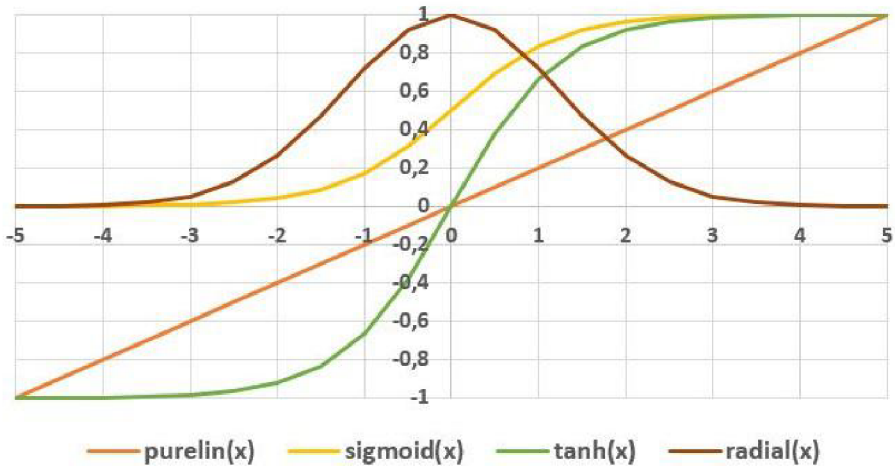
$$y = S\left(\sum_{i=1}^{n} w_i x_i + \phi\right) \tag{3.2}$$

, kde:

- y značí výstup neuronu,
- $S(x)$ je přenosová neboli aktivační funkce neuronu,
- x_i značí n vstupů neuronů,
- w_i značí synaptické váhy vstupů neuronů, kterými se násobí vstupní hodnoty x_i,
- ϕ značí práh.

Tento neuron má obecně n vstupů, reprezentujících dendrity biologického neuronu. Vstupům jsou přiřazeny váhy (např. float čísla), určující jejich propustnost. Tyto váhy

mohou být záporné (inhibice) nebo kladné (excitace). Dále má formální neuron práh ϕ, který určuje prahovou hodnotu jeho aktivace, při jejímž překonání neuron vyšle signál na svůj výstup Y v podobě přenosové neboli aktivační funkce $S(x)$. Vnitřním potenciálem nebo také aktivační hodnotou neuronu nazýváme sumu $\sum_{i=1}^{n} w_i x_i$ [9].



Obrázek 3.3: Nejčastěji používané aktivační funkce. Přejato a upraveno z [17].

Přenosové funkce mívají typicky tvar sigmoidy, ale mohou to být například i po částech lineární funkce nebo skokové funkce. Příklady často používaných přenosových funkcí se nachází na obrázku 3.3. Často jsou tyto funkce monotónní rostoucí, spojité, diferencovatelné a omezené. MCP model používá ještě váhu w_0, umožňující přičtení hodnoty navíc k sumě před výpočtem přenosové funkce. Tato hodnota v MCP modelu představuje aktivační práh neuronu. Nelineární neurony ale aktivační práh nepoužívají – pracují totiž jako přizpůsobitelné jednotky, které umožňují aproximovat trénovací data pomocí zvolené nelineární funkce, což již příliš neodpovídá funkcionalitě biologických neuronů [17].

3.4 Typy neuronových sítí a jejich výhody a nevýhody

Neuronové sítě lze rozčlenit do více typů, vhodných pro různé úlohy. Lze je rozdělit například podle vstupu na sítě s binárním vstupem a se spojitými vstupy. Oba tyto typy přitom mohou využívat přístup učení s učitelem nebo bez učitele. Neuronové sítě lze ale dělit také na základě jejich struktury, a to na strukturované a nestrukturované, přičemž poslední zmíněné se dále dělí na kompetiční a hierarchické modely. Kompetiční modely obsahují vrstvy, které mohou vykonávat různé funkce, vrstvy hierarchických modelů vykonávají všechny stejné funkce [20].

Síť, obsahující jediný neuron, dokáže rozdělit prvky množiny do dvou disjunktních množin pomocí proložení nadroviny n-rozměrným prostorem, kde n značí počet vstupů. Složitější úkoly zvládne spíše vrstevnatá neuronová síť, jejíž neurony jsou rozděleny do více typů vrstev – vstupní, výstupní a 1 či více skrytých. Sítě s menším počtem vrstev se učí rychleji, ty s větším počtem dokáží zase lépe zobecňovat. Počet neuronů v síti se může podle složitosti úkolu lišit od několika až po tisíce.

Vstupní vrstva není složena přímo z plnohodnotných neuronů, ale sestává z hodnot v datovém záznamu, který tvoří vstupy do další vrstvy neuronů. Ve výstupní vrstvě se pak nachází vždy jeden uzel pro každou klasifikační třídu. Při průchodu sítí je výsledkem přiřazení hodnoty každému výstupnímu uzlu a záznam je kategorizován do třídy, jejíž uzel má nejvyšší hodnotu.

Počet vrstev a neuronů v každé vrstvě je velmi důležitý. Co se týče rozložení vrstev a neuronů, není obecně možné vytvořit jedinou nejlepší síť pro daný typ aplikace. Existují pouze obecná pravidla pro tvorbu sítě [42].

Neuronové sítě mají mnoho výhod, mezi něž patří relativně jednoduchá implementace, možnost paralelního zpracování, schopnost učení a generalizace, schopnost přizpůsobení se a také distribuovaná reprezentace a výpočet. Sítě jsou nelineárním modelem, který je dobře srozumitelný v porovnání se statistickými metodami. Jejich konvergence je sice pomalá, ale v podstatě zaručená při použití kvalitního datasetu.

Sítě mají samozřejmě také své nevýhody, mezi něž patří například to, že je nelze později při přidání dodatečných tříd dat snadno přetrénovat – často je nutné začít trénink od začátku. Nevýhodou může být i fakt, že sítě pracují s „black box“ přístupem, takže jejich vnitřní funkcionalita není přímo známá, a také jsou velmi náročné na výpočetní výkon ve fázi tréninku. Ačkoliv síť může vynikat v řešení konkrétního problému, na který je natrénována, nedokáže intuitivně řešit jiné, podobné problémy. Neuronové sítě jsou také vždy pouze aproximací požadovaného řešení, takže je nutné počítat s chybou.

Uplatnění neuronových sítí v oblasti umělé inteligence je velmi široké. Obecně se tyto sítě využívají pro klasifikaci zadaných vstupů do tříd a také pro regresi (aproximaci). Konkrétněji je lze využít například pro rozpoznávání objektů, písmen, číslic, hledání shody mezi neznámými objekty nebo pro aproximaci funkce na základě jejích vzorků [8]. Dalšími oblastmi využití jsou například letecká doprava, filtrace EKG signálu či sféra finančnictví (predikce změny kurzů měn či akcií). Je také možné naučit síť reagovat na podněty od učitele (experta), kdy po naučení síť dokáže zpracovat i jiné, dříve neznámé podněty. Těchto schopností lze využít například v pivovarech či atomových elektrárnách [25].

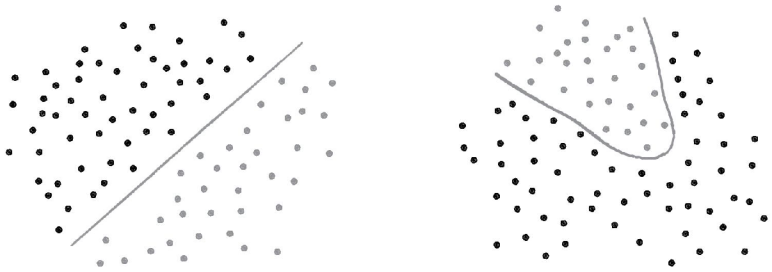
3.5 Perceptron a dopředná neuronová síť

Perceptron je obecný model nejjednodušší neuronové sítě, vytvořený již roku 1957 Frankem Rosenblattem. Tento výpočetní model obsahuje jediný neuron. Perceptron je také základním stavebním kamenem složitějších neuronových sítí. Skládá se z 1 nebo více vstupů, těla a jediného výstupu. Jedná se o dopředný model, kdy jsou zpracovány vstupy, poslané do neuronu, a výsledek je zaslán na výstup.

Učení perceptronu sestává z předložení několika trénovacích vzorků a výpočtu výstupů pro každý z nich. Po každé této iteraci jsou váhy w upraveny tak, aby byla minimalizována výstupní chyba, která je definována jako rozdíl mezi cílovými a skutečnými výstupy. Existují i různé další chybové funkce, jako například střední kvadratická odchylka, ale základní princip učení zůstává stejný [45].

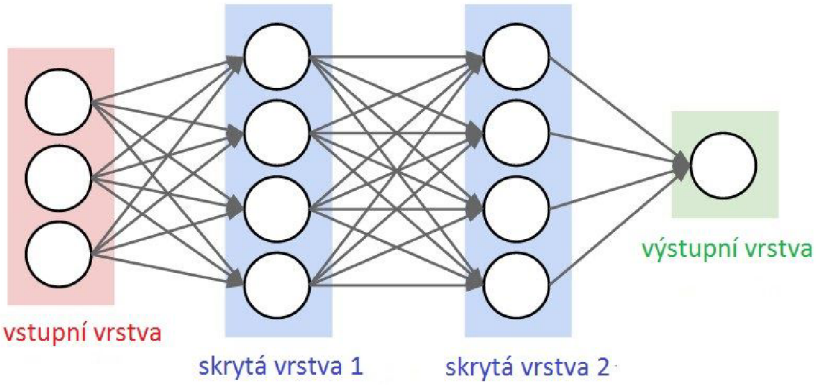
Velkým nedostatkem modelu perceptronu je jeho velice omezená použitelnost – dokáže totiž řešit pouze lineárně separovatelné problémy (takové, kde je možné data klasifikovat do tříd, oddělitelných pomocí přímky).

Na obrázku 3.4 lze vidět 2 případy – vlevo lineárně separovatelný problém, vpravo není již možné data separovat lineárně. Příkladem jednoduchého lineárně separovatelného problému je tabulka pro základní logickou funkci AND nebo OR. XOR ale není lineárně separovatelný problém, takže na něj jediný perceptron nestačí.



Obrázek 3.4: Lineárně separovatelný a neseparovatelný problém [41].

Pro složitější problémy je třeba použít vícevrstvý perceptron, také nazývaný dopředná neuronová síť (angl. *feed-forward*). Tato síť sestává z více perceptronů, propojených dohromady, které vytvářejí účinnější učicí mechanismus. Síť má typicky vstupní, výstupní a 1 nebo více skrytých vrstev. V příkladu na obrázku 3.5 je síť, která obsahuje jednu vstupní vrstvu o 3 jednotkách, výstupní o 1 jednotce a 2 skryté o 4 jednotkách. Každou jednotku tvoří perceptron. Jednotky vstupní vrstvy slouží jako vstupy pro jednotky první skryté vrstvy; jednotky druhé skryté vrstvy jsou vstupy pro výstupní vrstvu. Skrytá vrstva neuronové sítě je místem, kde síť ukládá svou interní abstraktní reprezentaci trénovacích dat [17]. Každé propojení mezi 2 neurony má váhu w (podobně jako u perceptronu). Každá jednotka vrstvy t je typicky propojena s každou jednotkou předchozí vrstvy $t-1$. Pro zpracování vstupních dat předložíme vstupní vektor vstupní vrstvě – to se provádí nastavením hodnot vektoru jako výstupů pro každou ze vstupních jednotek. Síť z obrázku 3.5 dokáže zpracovat třírozměrný vstupní vektor, protože je tvořena 3 vstupními jednotkami. Hodnoty vstupního vektoru jsou poté propagovány dopředu přes síť do skrytých jednotek za použití přenosové funkce váženého součtu pro všechny skryté jednotky, které v návaznosti spočítají své výstupy (aktivační funkce). Výstupní vrstva vypočítá své výstupy podobně jako skrytá vrstva. Výsledek výstupní vrstvy představuje výstup sítě [45].



Obrázek 3.5: Dopředná neuronová síť. Přejato a upraveno z [24].

Důležitou součástí neuronové sítě je tzv. aktivační funkce (jinak označovaná také jako prahová funkce nebo funkce přechodu). Jedná se o funkci, kterou každý perceptron posílá na výstup, pokud je překročena jeho vlastní prahová hodnota neboli potenciál [44]. Většina neuronových sítí používá nelineární aktivační funkce jako je například hyperbolický tangens

či sigmoida. Pouze s použitím lineární aktivační funkce by totiž vždy byl výsledek sítě lineární kombinací vstupů, upravených pouze pomocí vah. Bez nelineárních aktivačních funkcí by tedy byla dopředná neuronová síť prakticky stejně silná jako perceptron samotný, protože by se vždy mohla naučit pouze funkce, které jsou lineární kombinací jejích vstupů [41].

3.6 Backpropagation algoritmus a učicí problém

Nejčastěji používaný algoritmus hlubokého učení pro vícevrstvý perceptron se označuje termínem Backpropagation. Jedná se o algoritmus, využívající metody učení s učitelem, který pracuje se zpětnou propagací chyby. Ke své správné funkci vyžaduje diferencovatelnou aktivační funkci. Cílem je nalezení lokálního minima chybové funkce E, které probíhá pomocí metody gradient descent. Základní podoba algoritmu je následující:

1. Předložení trénovacího vzorku a jeho propagace dopředu skrz síť přes všechny neurony a vrstvy až do výpočtu výsledků výstupní vrstvou.

2. Aktualizace synaptických vah w_i na výstupech neuronů v průběhu zpětné propagace výstupní chyby (často používaná je střední kvadratická odchylka, udávaná jako $E = (1/2) * (t - y)^2$, kde t je cílová hodnota a y je skutečný výstup sítě).

Síť je nejdříve inicializována náhodně zvolenými váhami. Poté se vypočítá gradient chybové funkce a ten se použije pro úpravu původních vah. Úkolem je počítat rekurzivně tento gradient.

Podle obecné definice lze považovat dopřednou neuronovou síť také za výpočetní graf, jehož uzly představují výpočetní jednotky a orientované hrany posílají numerickou informaci mezi uzly. Každá tato výpočetní jednotka je schopná vyhodnocení jediné primitivní funkce svého vstupu. Síť tedy představuje řetězec funkčních kompozicí, které transformují vstup na výstupní vektor, nazvaný vzor (angl. *pattern*), tvořený uspořádanou dvojicí vstupního a výstupního vektoru. Síť je tedy od vstupní po výstupní vrstvu implementací určité složené funkce (angl. *network function*). Problém učení tedy znamená nalezení optimální kombinace vah jednotlivých hran tohoto výpočetního grafu tak, aby síťová funkce ϕ aproximovala danou funkci f co nejlépe. Při trénování ale není známa funkce f explicitně, ale pouze implicitně prostřednictvím příkladů.

Jako chybová funkce se u dopředných vícevrstvých neuronových sítí často využívá střední kvadratická odchylka. Chybová funkce značí míru naučenosti sítě [39].

3.7 Učení neuronových sítí

Schopnost učení je základním rysem inteligence. Místo následování sady pravidel, specifikovaných člověkem, se síť sama učí příslušná „skrytá pravidla“ (např. vztahy mezi vstupy a výstupy) na základě reprezentativních příkladů. Tato schopnost je pro určité úkoly jednou z velkých výhod sítí oproti tradičnímu programování.

V kontextu neuronových sítí je proces učení chápán jako obnovování síťové architektury, prahových funkcí a vah, náležejících jednotlivým spojením mezi neurony tak, aby mohla síť efektivněji vykonávat požadovaný úkol. Učení většinou probíhá jako opakované předkládání trénovacích vzorů z trénovací množiny, přičemž výkon sítě se zlepšuje iterativním obnovováním jejích vah s cílem minimalizovat chybovou funkci E. Epocha je označení pro

každé jedno předložení celé množiny trénovacích vzorů síti a jejich průchod sítí. Trénink obvykle trvá mnoho epoch, s jejichž rostoucím počtem je v ideálním případě síť stále lépe natrénovaná.

Proces učení vede k těmto třem událostem:

1. Neuronová síť je stimulována svým okolním prostředím.
2. Výsledkem této stimulace jsou změny ve výše popsaných parametrech sítě.
3. Díky změnám, provedeným ve vnitřní struktuře neuronové sítě tato reaguje novým způsobem na okolní prostředí.

Učicí algoritmus lze pak definovat jako sadu korektně definovaných pravidel pro řešení učicího problému. Neexistuje univerzální učicí algoritmus, který by byl vhodný pro všechny neuronové sítě; různé algoritmy se od sebe liší například ve způsobu, jakým je formulována úprava synaptických vah neuronů. Dalším důležitým faktorem je způsob, jakým je každá neuronová síť propojena se svým okolím. V tomto kontextu se hovoří o učicím paradigmatu, které se vztahuje k modelu prostředí, ve kterém neuronová síť funguje [19].

Po dokončení trénování (v některých případech i v průběhu učícího procesu mezi epochami) se ověřuje míra naučenosti neuronové sítě pomocí testovací množiny. Tato množina je postupně předkládána neuronové síti a poté se na základě zadaného a očekávaného vstupu vypočítá testovací chybová funkce *Etest*. Trénování sítě vždy probíhá až do dosažení předem stanovené podmínky, kterou může být například dosažení minima hodnoty *Etest*, předem stanoveného počtu trénovacích epoch nebo dosažení minimální hodnoty chybové funkce E, která je předem daná [51].

3.8 Konvoluční neuronové sítě

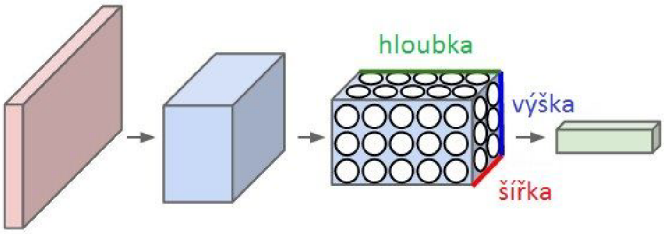
Konvoluční neuronové sítě (angl. *Convolutional Neural Networks*, zkratka *CNN*), které jsou hlavním předmětem této práce, jsou variantou vícevrstvých neuronových sítí. Jedná se o specifickou třídu dopředných sítí, jež jsou v praxi často využívané ke klasifikaci obrazových dat nebo videa. Výstupy vrstev se místo použití aktivační funkce získávají konvolucí vstupu a k němu příslušejícího konvolučního jádra. Některé vrstvy konvolučních sítí jsou tvořeny trojrozměrnými poli neuronů a na rozdíl od klasické plně propojené neuronové sítě jsou zde neurony ve vrstvě propojeny vždy pouze s malou částí předchozí vrstvy.

Konvoluční neuronové sítě mohou mít různou architekturu; v praxi používané algoritmy jsou ale velmi podobné pro všechny z nich. Konvoluční neuronová síť se kromě vstupní a výstupní vrstvy skládá ještě z následujících specifických typů vrstev:

- Konvoluční vrstva sestává z obdélníkové mřížky neuronů a vyžaduje, aby jí předcházející vrstva byla také takovouto mřížkou. Každý neuron dostává na vstup data z obdélníkové podsekce předchozí vrstvy; váhy pro tuto sekci jsou stejné pro každý neuron v konvoluční vrstvě. Konvoluční vrstva je tedy obrazem konvoluce předchozí vrstvy, kde váhy specifikují konvoluční filtr. Součástí konvoluční vrstvy může být více těchto mřížek, kdy každá z nich dostane vstupy od všech mřížek předchozí vrstvy. Vrstvy mohou používat různé filtry.
- Vrstva poolingu se může nacházet za každou konvoluční vrstvou. Jejím úkolem je brát malé, obdélníkové bloky z konvoluční vrstvy, a sloučit je do jediného výstupu z tohoto bloku – jedná se o podvzorkování konvolučního obrazu na vstupu. Pomocí poolingu

se dá snižovat počet parametrů a množství výpočtů v síti, čímž lze také předcházet riziku přeučení sítě (angl. *overfitting*). Pooling lze provádět různými metodami, jako je například průměr nebo maximum.

- Klasifikační vrstva neboli plně propojená vrstva je nejvyšší vrstvou konvoluční sítě a odehrává se v ní vysokoúrovňové rozhodování. Počet jejích neuronů odpovídá počtu klasifikačních tříd. Vstupem všech jejích neuronů jsou všechny výstupy z předchozí vrstvy. Těmito výstupy jsou příznakové vektory vstupního obrazu. Plně propojená vrstva vezme výstupy všech neuronů z předchozí vrstvy a spojí je do jediného výstupu [3].



Obrázek 3.6: Vrstvy konvoluční neuronové sítě. Přejato a upraveno z [24].

Základní struktura konvoluční neuronové sítě může vypadat jako na obrázku 3.6. Tato síť obsahuje vrstvy s třírozměrným uspořádáním neuronů. Každá vrstva konvoluční sítě převádí trojrozměrný vstupní vektor na výstupní vektor prostřednictvím neuronových aktivací. V tomto případě obsahuje červená vstupní vrstva zpracovávaný obrázek, takže výška a šířka této vrstvy odpovídají rozměrům obrázku. Hloubka u skryté vrstvy `depth=3` je dána počtem barevných kanálů (zde RGB) [24].

Vrstvy obsahují různé počty příznakových map. Příznaky mohou být například různé kombinace barev na různých místech obrazu – u příkladu krajiny by horní část obsahovala modrou barvu a dolní část spíše zelenou.

Jednodimenzionální konvoluční neuronové sítě lze využít například v analýze zvuku. Obecně je ale možné pracovat i nad vícedimenzionálními daty – například v rozpoznávání obrazu se používají 2D konvoluční neuronové sítě, kde se pracuje s *patches* místo se segmenty. 3D konvoluční sítě se někdy používají pro objemová data (lékařské snímky) nebo video. Nejsou ale zatím široce využívané a jejich architektura je obtížně vizualizovatelná [35].

Kapitola 4

Existující knihovny, frameworky a klient-server aplikace

Pro tuto práci jsem nejdříve provedla průzkum existujících řešení se zaměřením na mobilní klient-server aplikace, nabízející možnosti rozpoznávání objektů v obraze. V této části jsou popsány některé z nich. V dalším kroku jsem se zjišťovala, zda pro neuronové sítě existuje nějaký dobře použitelný framework nebo knihovna, aby nebylo nutné veškeré potřebné algoritmy a datové struktury vytvářet od začátku. Některé z těchto knihoven a frameworků jsou popsány v sekci 4.2.

4.1 Existující klient-server aplikace pro mobilní telefony

V mobilních zařízeních (především ve smartphonech, vybavených fotoaparátem a kamerou) je v současné době díky jejich rostoucím výpočetním schopnostem možné využívat například rozšířenou realitu. Ta přidává do obrazu skutečného světa různé další prvky, jako jsou informace o objektu nebo pokyny navigace. Takováto aplikace může na straně serveru rozpoznávat objekty a na straně klienta je sledovat. Je možné rozpoznávat stacionární objekty, jako jsou například budovy, i nestacionární objekty (obaly médií). Základní koncept rozšířené reality se skládá z identifikace objektů reálného světa, jejich sledování a poté rozšiřování scény umělými objekty. Databáze se typicky nachází na straně serveru a potenciální omezení spočívá v její velikosti [41]. Rozšířenou realitu používají aplikace jako Layar [26] nebo CamFind [7], zmíněné níže. Může sloužit také pro hry, a to i takové, kdy se hráč musí pohybovat po městě (např. aplikace Ingress od Googlu).

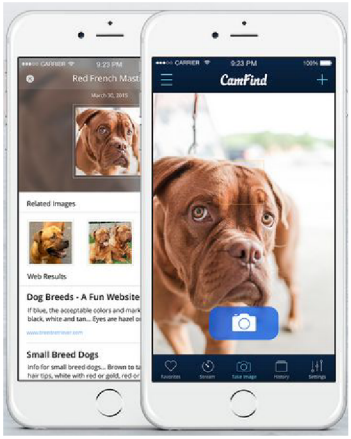
Využití detekce a identifikace objektů v obraze u mobilních zařízení je mnohem více. Další aplikace mohou sloužit k identifikaci produktů, knih, souhvězdí na obloze či významných míst v obraze. Následuje popis vybraných mobilních aplikací, zaměřených na detekci a rozpoznání objektů v obraze.

- **Layar** je aplikace, umožňující propojení reálného světa a digitálního obsahu. Zvládá jak běžné rozpoznávání budov, tak i přidávání videí a animací do článků v časopisech a do různého digitálního obsahu – tato vlastnost jej dělá zajímavým i pro využití v oblasti reklamy. Layar také nabízí uživatelům možnost vytvořit si vlastní rozšíření reality přidáním vlastních videí nebo animací do obrázků, letáků, plakátů, balení produktů a podobně, a vzniklý obsah pak sdílet na sociálních sítích. Na internetu je dostupné velké množství materiálů, které Layar dokáže zpracovat a „rozpohybovat“. Z běžných

objektů jej lze využívat kupříkladu k rozpoznání památek a dalších význačných bodů ve městech. Způsob práce s aplikací je zobrazen na obrázku 4.1.

Obrázek 4.1: Aplikace Layar [26].

- **Camfind** je mobilní aplikace, která dokáže rozpoznávat objekty v obraze – jedná se o zajímavou alternativu textového vyhledávání. Objekt je nejprve vyfocen a poté CamFind využije technologii vizuálního vyhledávání. Výsledky vyhledávání lze sdílet s přáteli. CamFind je možné využít například pro vyhledávání produktů v obchodě, pro které aplikace srovnává ceny, nebo ke získání informací při zaměření objektivu na filmové plakáty, restaurace (zobrazení menu podle loga) apod. Poznávat ale lze také zvířata nebo budovy. Jako výsledky vyhledávání jsou nabízeny související obrázky, případně možnosti nákupu vyhledávaného produktu v okolí. Okno aplikace je možné vidět na obrázku 4.2.

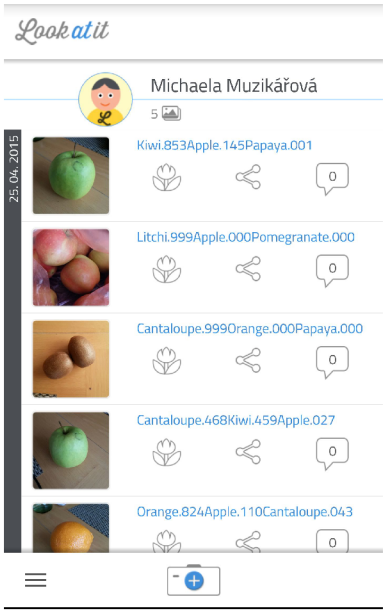


Obrázek 4.2: Aplikace CamFind [7].

- **LookTel Recognizer** je aplikace, jež má sloužit především zrakově postiženým lidem jako pomocník při rozpoznávání objektů denní potřeby (balené zboží, plechovky s nápoji, peníze, CD, nápisy, cedule, průčelí obchodů a další). Na začátku je nutné

aplikaci naučit rozpoznávat požadované objekty, které se musí nasnímat a pojmenovat. Tuto fázi může pro zrakově postiženého udělat asistent, nebo lze stáhnout hotové knihovny objektů, které mezi sebou sdílí uživatelé. Rozpoznávání v této aplikaci probíhá v reálném čase bez nutnosti fotografování objektu. Jakmile je ve scéně detekován objekt z databáze uživatele, aplikace uživatele na tuto skutečnost upozorní přehráním namluveného textu k rozpoznanému objektu [28].

- **LookAtIt** [29] je mobilní aplikace, jejímž účelem je sdílení fotogalerií, obsahujících například nakupované zboží nebo jiné objekty. Umožňuje také chat mezi uživateli za účelem rychlé rady při nákupu. Vyfocené objekty je možné řadit do kategorií, sdílet s přáteli, přidávat k nim popisky, informace o ceně a další. Tato aplikace nenabízí funkce detekce a rozpoznávání objektů v obraze, ale byla využita jako klientská část aplikace, vzniklé v rámci této bakalářské práce. Okno aplikace s několika fotografiemi uživatele je možné vidět na obrázku 4.3.



Obrázek 4.3: Aplikace LookAtIt [29].

4.2 Existující frameworky a knihovny pro hluboké učení

V současné době existuje velké množství knihoven i ucelených frameworků pro hluboké učení, které lze využít pro účely detekce a rozpoznání objektů v obraze. Některé z nich jsou vhodné pro specifické úkoly, jiné nabízejí obecnější možnosti využití. Knihovny a frameworky se mezi sebou liší také implementačním jazykem a licencí, pod kterou jsou distribuovány. Následuje stručný popis těch, které byly uvažovány pro tuto bakalářskou práci:

- **OpenDeep** [37] je knihovna pro hluboké učení, využitelná pro komerční i výzkumné účely. Tato knihovna je vhodná pro obecné použití a je napsána v jazyce Python. Nabízí jak předpřipravené modely, tak i možnost vytvoření vlastních, včetně podpory

výpočtů na GPU a CuDNN[1].

- **Torch7** [43] je open-source framework s rozhraním, podobným Matlabu. Místo obvyklého Pythonu ale používá skriptovací jazyk Lua. Umožňuje paralelizaci s pomocí OpenMP[2] a CUDA[3]. Dále poskytuje rozhraní pro jazyk C přes LuaJIT kompilátor. Tento framework lze využít také pro vestavěné systémy, včetně možností paralelizace za pomocí CPU a GPU.

- **DeepLearning4j** [10] je open-source knihovna hlubokého učení, napsaná pro jazyky Java a Scala. Tato knihovna je zamýšlena především pro obchodní prostředí. Umožňuje snadné vytváření prototypů nejen lidem s hlubokými znalostmi oboru. Nabízí různé typy neuronových sítí včetně konvolučních a lze ji využít například k rozpoznávání tváří, hlasovému vyhledávání nebo přepisu řeči do textu. Umožňuje běh na GPU.

- **Matlab Image Processing Toolbox** [31] je komerční produkt firmy MathWorks, nabízející algoritmy a funkce pro zpracování a analýzu obrazu, vizualizaci a možnosti tvorby vlastních algoritmů. Toolbox umožňuje různé druhy analýzy obrazu, segmentaci obrazu, metody vylepšení vlastností obrazu jako například redukci šumu nebo geometrické transformace. Mnoho z těchto funkcí podporuje vícejádrové procesory a také běh na GPU. Toolbox dokáže pracovat s různými typy obrázků včetně HDR snímků nebo tomografických dat.

- **OpenCV** je open-source softwarová knihovna, implementovaná v C++, která se zaměřuje na počítačové vidění a strojové učení. Knihovna obsahuje přes 2500 optimalizovaných algoritmů počítačového vidění a strojového učení. Tyto algoritmy mohou být použity například pro detekci a rozpoznání obličejů, identifikaci objektů, klasifikaci akcí člověka ve videích, sledování pohybu kamery nebo očí, spojování fotografií pro vytvoření panoramat ve vysokém rozlišení a další. OpenCV je využíváno jak velkými firmami, tak různými startupy. Poskytuje rozhraní pro C++, C, Python, Javu a MATLAB.

- **Caffe** [21] je open-source framework pro hluboké učení s podporou GPU včetně technologií CUDA a CuDNN. Framework je napsaný v C++, ale lze jej použít i ve spolupráci s Pythonem a MATLABem. Byl vyvinut především pro úkoly, týkající se počítačového vidění, a pro většinu z nich využívá konvoluční neuronové sítě. Caffe dosahuje velmi vysokých rychlostí při zpracování obrazů – například s jedinou grafickou kartou typu NVIDIA K40 zvládne zpracovat přes 60 milionů obrázků za den (1 ms/obraz při inferenci a 4 ms/obraz při učení). Framework zahrnuje nejmodernější algoritmy hlubokého učení a také kolekci referenčních modelů. Oddělením reprezentace modelu od samotné implementace usnadňuje experimentování s modelem a také přechody mezi platformami (například pro nasazení v cloudovém prostředí). Caffe je využíván mnoha velkými výzkumnými projekty, ale také různými startupy v oblastech počítačového vidění, zpracování řeči a multimédií.

[1] Knihovna primitiv pro neuronové sítě, akcelerovaná pomocí GPU.

[2] API pro podporu paralelního programování.

[3] Technologie firmy NVIDIA, umožňující využití GPU pro běh programů, zapsaných v C, C++ nebo Fortranu.

4.3 Caffe framework – formát modelu a solver

Caffe framework definuje neuronovou síť za pomocí vlastního modelového schématu, jehož základem jsou vrstvy a spojení mezi nimi. S informacemi pracuje ve formě blobů, představujících jednotné rozhraní paměti. Bloby jsou reprezentovány čtyřrozměrným polem, které tvoří obal okolo zpracovávaných dat a také zajišťuje synchronizaci mezi CPU a GPU. Toto pole v sobě nese data a jejich deriváty v podobě dávek nebo obrázků a také parametry modelu.

Rozměry každého blobu jsou $N * K * H * W$. Číslo N udává velikost zpracovávané dávky dat. Kanál K je dimenze příznaků, např. pro RGB obrázky K bude 3. Parametry pro každý blob se mohou lišit podle typu vrstvy – například pro konvoluční vrstvu s 96 filtry o rozměrech $11 * 11$ a třech vstupech bude mít blob rozměry $96 * 3 * 11 * 11$.

Vrstvy nabízí velké množství operací, obsahují filtry, pool, přijímají vnitřní produkty sítě, aplikují různé transformace, normalizují, načítají data a počítají ztrátu loss. Vrstva přijímá vstup pomocí spodních propojení a výstup posílá horními propojeními. Každá vrstva využívá tři důležité výpočty: `setup`, `forward` a `backward`. `Setup` značí inicializaci vrstvy a jejích spojení, které probíhá při počáteční inicializaci modelu. Operace `forward` počítá výstup na základě vstupu zdola a předává jej vyšší vrstvě. U operace `backward` je dán gradient s ohledem na horní výstup a počítá se gradient s ohledem na vstup, který se zasílá níže. Vrstva spočítá gradient s ohledem na své parametry a uloží si jej.

V modelu jsou implementovány dvě různé `Forward` a `Backward` funkce, každá ve variantě pro CPU i GPU.

Vrstvy mají 2 hlavní úkoly, významné pro funkci sítě: dopředný krok, který vezme vstupy a vyprodukuje výstupy, a zpětný krok, který vezme gradient s ohledem na výstup a spočítá gradient s ohledem na parametry a na vstupy, které jsou zpětně propagovány do předchozích vrstev. Tyto kroky jsou kompozicí dopředných a zpětných kroků každé vrstvy.

Síť definuje funkci a její gradient pomocí kompozice všech výstupů každé vrstvy. Zpětná kompozice pak počítá gradient za použití loss funkce.

Síť je množina vrstev, propojených do výpočetního grafu – jedná se o orientovaný acyklický graf. Caffe se přitom stará o veškeré činnosti, vyžadované pro korektnost všech dopředných a zpětných kroků. Síť začíná vstupní datovou vrstvou, která načítá data z disku, a končí výstupní loss vrstvou, která počítá cíl úkolu, kterým může být například klasifikace.

Síť je definována jako množina vrstev a jejich propojení v modelovacím jazyce. Inicializace modelu, spouštěná pomocí funkce `Net::Init()`, sestavuje orientovaný acyklický graf vytvořením blobů a vrstev a volá funkci `SetUp()` pro vrstvy. `Net::Init()` má na starosti také další úkoly, jako je například ověření korektnosti architektury sítě.

Konstrukce sítě je nezávislá na platformě – bloby a vrstvy skrývají implementační detaily před definicí modelu. Po sestavení může síť běžet na CPU nebo na GPU – to lze ovlivnit pomocí nastavení přepínače, definovaného v `Caffe::mode()`, který se nastavuje funkcí `Caffe::setmode()`. Vrstvy mají odpovídající CPU a GPU rutiny, které produkují identické výsledky.

Model je definován v `prototxt` – `plaintext protocol buffer` schématu. Vytvořené a naučené modely jsou serializovány jako binární `caffemodel` soubory. Formát modelu je definován pomocí `protobuf` schématu v `caffe.proto`. Caffe používá Google Protocol Buffer kvůli efektivní serializaci, srozumitelnému textovému formátu, který je kompatibilní s binární verzí, a také možnosti efektivní implementace rozhraní v mnoha různých programovacích jazycích. Toto vše přispívá k flexibilitě a rozšiřitelnosti modelů, vytvářených za použití frameworku Caffe [4].

Solver modelu zajišťuje optimalizaci, řeší aktualizace parametrů sítě, periodicky vyhodnocuje míru naučenosti sítě a vytváří „snímky“ (angl. *snapshot*) aktálního stavu modelu a solveru. Dále vytváří trénovací síť pro učení a testovací síť či sítě pro vyhodnocování. Při kadé iteraci se počítá výstup, ztráta sítě a gradienty. Gradienty jsou použity k aktualizacím parametrů. Stav solveru se aktualizuje na základě akuální míry naučenosti sítě, historie a použité metody [6].

Kapitola 5

Realizace

V této kapitole je nejprve popsán postup výběru prostředků, použitých v rámci bakalářské práce. Následují informace o návrhu systému včetně jeho diagramu. Kapitola pokračuje informacemi o realizaci jednotlivých součástí aplikace. Následují údaje o postupu trénování sítě a o použitém datasetu.

5.1 Analýza současného stavu

V rámci této bakalářské práce bylo hlavním cílem vytvoření klient-server aplikace s využitím vhodné existující mobilní aplikace, která by jako celek dokázala rozpoznávat vybrané objekty.

Na začátku práce bylo třeba provést důkladnou analýzu literatury a existujícího softwaru se zaměřením na rozpoznávání objektů v obraze a na tvorbu mobilních klient-server aplikací. Dalším krokem byla volba vhodného typu objektu pro rozpoznávání a také vhodné detekční a rozpoznávací metody pro vybraný objekt. Jako objekt jsem na základě analýzy softwaru podobného zaměření a po zvážení možných uplatnění aplikace zvolila ovoce s cílem, že by aplikace do budoucna při vhodném rozšíření mohla sloužit zrakově postiženým lidem jako pomocník při nakupování (například u tvarově podobného ovoce).

Poté jsem vybírala detekční a rozpoznávací metodu. Nejdříve jsem uvažovala o boosting algoritmech, popsaných v kapitole 3, ale nakonec jsem zvolila neuronové sítě, protože mne tato metoda nejvíce zaujala. Dalším důvodem bylo, že neuronové sítě, zmíněné v kapitole 3, jsou velice dobře použitelné pro rozpoznávání objektů v obraze. Hodí se totiž právě pro úkoly, které není snadné popsat a vyřešit algoritmicky. Především konvoluční neuronové sítě, zmíněné v sekci 3.8, mají velmi výhodné vlastnosti pro rozpoznávání objektů v obraze. Jsou také relativně jednoduché na vytváření s využitím frameworku.

Pro usnadnění návrhu a tvorby aplikace jsem provedla průzkum dostupných frameworků a knihoven, využitelných pro detekci objektů v obraze, jejichž podrobnější popis se nachází v kapitole 4. Z těchto frameworků a knihoven, vhodných pro neuronové sítě, jsou dostupné jako open-source všechny kromě nástroje od firmy Matlab. V užším výběru jsem se rozhodovala mezi frameworky Caffe, Torch7 a OpenCV.

Na základě provedeného průzkumu současného stavu, zájmu o oblast neuronových sítí a zjištění dostupných možností jsem si k práci vybrala z dostupných frameworků a knihoven Caffe framework, protože podporuje konvoluční neuronové sítě, vhodné pro práci s obrazem, dále je open-source, má kvalitní dokumentaci a dosahuje prokazatelně vysokých rychlostí při zpracování obrazu. Obsahuje také předpřipravené modely, využitelné pro výzkum a

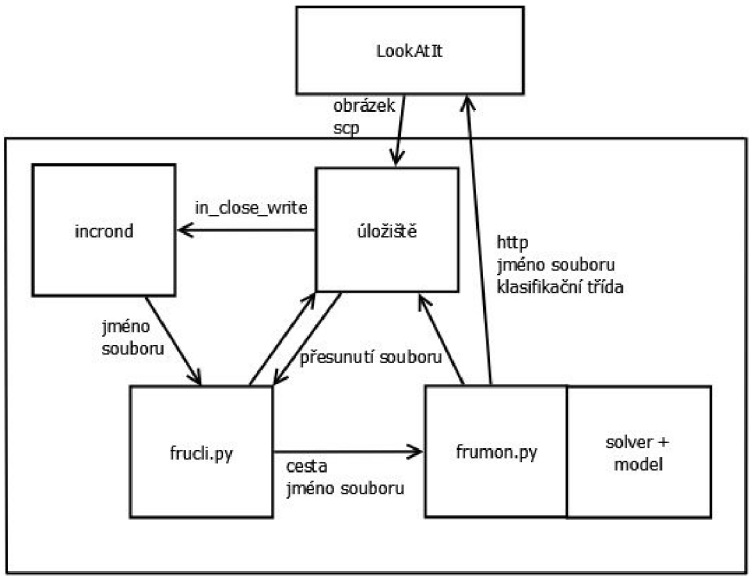
s možností jejich úpravy pro vlastní využití. Některé z těchto existujících modelů byly natrénovány na databázi projektu ImageNet [40], která byla využita i v rámci této práce. Caffe framework je také zaměřený cíleně právě na konvoluční neuronové sítě, narozdíl od OpenCV knihovny, vhodné pro širší využití.

Dalším krokem bylo vhodně navrhnout způsob implementace přenosu dat mezi klientem a serverem a připravit rozhraní serveru, nezbytné pro ověření aplikace. Mobilní část vytvořené aplikace by měla dokázat pořídit snímek za pomocí fotoaparátu smartphonu, a tento pak zaslat na server, kde dojde k dalšímu zpracování a jeho předložení neuronové síti. Nakonec se do mobilní aplikace ze serveru zašle odpověď s nejvíce pravděpodobnými klasifikačními třídami. Jako mobilní aplikace byla využita po domluvě s autory aplikace LookAtIt, upravená pro účel této práce a popsaná v části 4. Jádro bakalářské práce pak tvořila serverová část a návrh komunikace mezi klientem a serverem.

Poslední fázi práce tvořily experimenty s aplikací, zahrnující otestování úspěšnosti neuronové sítě při práci nad síti neznámou databází. Za tímto účelem jsem navrhla dva experimenty. V prvním z nich byla otestována míra úspěšnosti rozpoznávání druhů ovoce sítí za použití databáze, obsahující volně dostupné obrázky z databank. V další části testování probíhalo na obrázcích, které jsem vyfotila v obchodech a doma. Získané výstupy byly statisticky zpracovány a celý průběh a výsledky těchto experimentů jsou podrobněji popsány v kapitole 6. Kompletní výsledky experimentů lze najít na přiloženém DVD. Na závěr jsem se zamýšlela nad možnými rozšířeními aplikace do budoucna, kterých připadá v úvahu řada.

5.2 Návrh systému

Za použití vybraných výše zmíněných prostředků pro tvorbu systému bylo možné navrhnout klienta a server a dále stanovit způsob zasílání dat mezi nimi. Komunikaci mezi těmito částmi systému jsme po dohodě s autory mobilní aplikace LookAtIt navrhli tak, aby se přenášelo minimum informací a protokol byl co nejjednodušší. Její detaily jsou popsány níže.

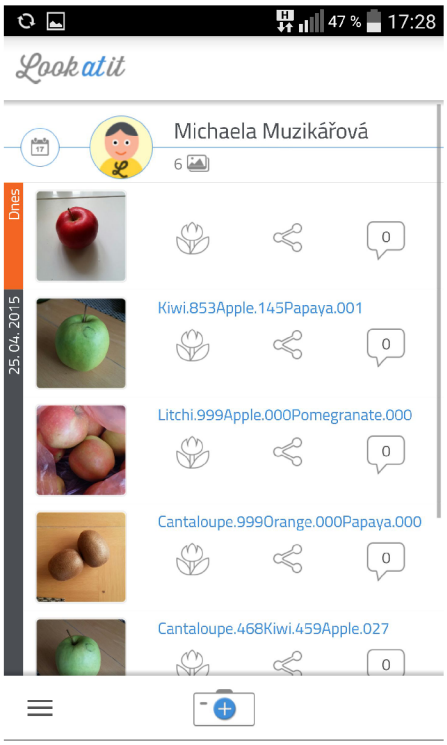


Obrázek 5.1: Diagram vytvořeného systému.

Celý vytvořený systém je rozdělen do 6 částí, které jsou zobrazeny na obrázku 5.1, kde lze také vidět, jaká data se posílají mezi různými součástmi systému. Podrobněji jsou všechny části systému popsány v následujících sekcích.

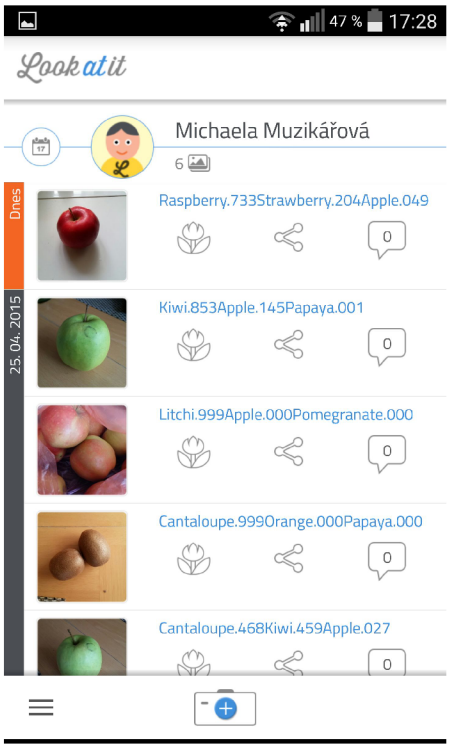
5.3 Klientská část aplikace

Po domluvě s autory aplikace LookAtIt, podrobněji popsané v sekci 4, byla tato aplikace s několika drobnými úpravami využita jako klientská část vytvářeného systému. Obrázky se v aplikaci řadí od nejnovějšího po nejstarší a jsou ihned po pořízení uživatelem aplikace, založeným pro účel této práce, automaticky nahrávány pomocí programu `scp` na server do složky `upload` pod unikátními názvy. Po klasifikaci obrázku jsou určené třídy v LookAtIt pro uživatele, který fotografii nahrál, zobrazeny v názvu příslušného souboru po jeho aktualizaci.



Obrázek 5.2: LookAtIt – nahrávání obrázku.

Na obrázku 5.2 je možné vidět průběh nahrávání první fotografie v pořadí na server (ikona nahrávání vlevo nahoře) ihned po vyfocení obrázku. Obrázek 5.3 ukazuje stav, kdy přišla ze serveru odpověď s třídami klasifikace. Do názvu obrázku se uloží jméno prvních 3 klasifikačních tříd a pro každou třídu její pravděpodobnost. Ta je pro každou třídu zobrazena ve formátu podílu z celku – výsledek 1.00 pro danou třídu znamená 100% pravděpodobnost. Aby se v aplikaci výsledek zobrazil správně, zaokrouhluje se na 1 desetinné místo.



Obrázek 5.3: LookAtIt – odpověď ze serveru s klasifikačními třídami.

5.4 Rozhraní a serverová část

Mobilní aplikace byla nastavena tak, že jakmile uživatel vloží do položky novou fotografii, je tato nahrána na server do adresáře `upload`. Jméno souboru je vždy unikátní v rámci systému. Ze serveru se poté informace odesílá zavoláním skriptu, vytvořeným autory aplikace LookAtIt, kde se URL adresa skládá z tokenu, který je pevně daný s neomezenou časovou platností, dále jména souboru s fotkou ze serveru a v sekci `data` je odesílaná informace. Skript vrací 0, pokud je vše v pořádku, nebo číslo chyby. Informace, odeslaná ze serveru klientovi se vkládá k příslušné položce jako její název. Pokud nastane problém při zpracování položky solverem, do názvu souboru se uloží textový řetězec „Chyba zpracování“. Poté může uživatel vytvořit novou fotografii a zkusit celý proces zopakovat.

Serverová část aplikace se skládá z několika podčástí. První z nich je úložiště příchozích souborů. Na straně serveru se používá utilita `incrond` [27] ke zjištění, zda do adresáře pro obrázky přibyl nový soubor. Do úložiště se ukládají nově vyfocené fotografie z aplikace LookAtIt. Obsahuje tři složky – `upload`, `queue` a `processed`. Po nahrátí nového souboru do složky `upload` se spustí skript `frucli.py` s parametrem, jímž je jméno souboru s jeho cestou. Tento skript soubor následně přesune do složky `queue` a skriptu `frumon.py` předá jeho cestu pomocí socketu. `Frumon.py` poté zavolá `solver` neuronové sítě pomocí `python wrapperu`. `Solver` obrázek zpracuje s použitím neuronové sítě a po tomto procesu vrátí síť 5 nejvíce pravděpodobných klasifikačních tříd spolu s příslušnými pravděpodobnostmi. Tyto třídy jsou démonem identifikovány (číslům klasifikačních tříd jsou přiřazeny názvy druhů ovoce) a v textové podobě názvu třídy jsou pak 3 nejpravděpodobnější třídy spolu s procenty pravděpodobnosti (zaokrouhlené na 1 desetinné místo) zasílány pomocí skriptu zpět do

aplikace LookAtIt. Čísla tříd jsou spojena s jejich názvy v souboru `synsetwords.txt`.

5.5 Průběh trénování neuronové sítě

Před prací na neuronové síti byl nejdříve na server nainstalován framework podle návodu, který je možné najít v příloze B. Poté byly vytvořeny výše popsané součásti aplikace. Dále bylo třeba vytvořit neuronovou síť, která se poté trénovala na připravených datech. Tato síť byla počítána automaticky v průběhu tréninku na základě zadaných parametrů.

Všechny obrázky datasetu byly nejprve zmenšeny pomocí skriptu `createimagenet.sh` (parametr `RESIZE=true`) na velikost 256x256, protože síť se s využitím menších, jednotně velkých obrázků dokáže rychleji a efektivněji učit. Rozměry obrázků navíc určují velikost vstupního vektoru a také počet neuronů vstupní vrstvy. Poté proběhl výpočet hodnoty `mean`. Tento výpočet je implementován v souboru `tools/computeimagemean.cpp`. Trénink je navíc možné po jeho přerušení znovu spustit z určitých bodů jeho postupu – tzv. „snapshotů“, které ukládají všechny potřebné informace pro obnovu daného stavu modelu včetně všech potřebných parametrů, čehož jsem několikrát využila při problémech se serverem.

Z předpřipraveých modelů, které framework Caffe nabízí, jsem využila konfiguraci z modelu ImageNet pro učení. Některé soubory se vytvářely automaticky při kompilaci Caffe a ostatní soubory pochází ze složky `/caffe/examples`, ve které se nachází příklady vytvořených modelů. Průběh učení sítě lze definovat nastavením parametrů v souboru `solver.protoxt`. Mezi tyto parametry patří například celkový počet iterací, krok učení, počet iterací, po kterém se aktuální stav sítě testuje na validačních datech a po kterých se vypisují do terminálu informace o aktuálním průběhu a také počet iterací, po kterém se vytváří snímky aktuálního stavu modelu. Hodnoty, které mají vliv na vlastní učení sítě, jsem experimentálně upravovala tak, aby učení sítě konvergovalo a ztráta `loss` nedosahovala hodnot `NAN` nebo `INF`. Původní nastavení těchto parametrů odpovídalo jejich nastavení v hotovém modelu Zoo [5].

Konečnými parametry pro síť byly tyto:

- Interval testu na validačních datech je 1000 iterací.
- `Learning rate` 0.002.
- Vytvoření snapshotu modelu a solveru po každých 10000 iteracích.
- Celkový počet 400000 iterací.
- Využití GPU pro výpočty.

Síť byla vytvářena na serveru s OS Ubuntu 14.04.2 LTS a trénována s využitím GPU. Původní grafická karta NVIDIA GeForce GT 610 ale nestačila, protože by s ní trénink trval příliš dlouho, a tak byla na serveru vyměněna za NVIDIA GeForce GTX 460, která podporuje také technologii cuDNN. Procesor serveru je Intel Core 2 Quad@2.4 GHz. Hotová natrénovaná síť je k dispozici na přiloženém DVD.

5.6 Dataset

Pro správné učení neuronové sítě je nutný kvalitní dataset. Pro neuronovou síť, vytvořenou za účelem této bakalářské práce, byla využita část datasetu z projektu ImageNet, vytvářeného pod hlavičkou Stanfordské univerzity. Databáze tohoto projektu obsahuje přes

15 milionů lidmi klasifikovaných obrázků, rozdělených do více než 22000 klasifikačních tříd. Z nich bylo vybráno 11 tříd, obsahujících požadované druhy ovoce, a odtud pochází všechna trénovací i validační data, využitá pro tvorbu a trénink sítě. Jelikož Caffe funguje tak, že po určitém počtu iterací v průběhu tréninku sítě ověřuje aktuální úspěšnost modelu pomocí validačních dat, bylo nutné hned na začátku práce sesbírat jak trénovací, tak i validační data.

Obrázek 5.4: Příklady obrázků z datasetu.

Trénovací vstupy jsou popsány v souboru `train.txt`, validační v souboru `val.txt`, a to pomocí názvů souborů s cestou a příslušné klasifikačních třídy. Trénovací data se nachází ve složce `imagenet/train`, kde je v podsložkách rozděleno ovoce podle druhů. Validační data se všechna společně nachází ve složce `imagenet/val`. V souboru `synset_words.txt` je popsáno mapování mezi podsložkami trénovacích dat a názvy klasifikačních tříd.

Druh ovoce	**Trénink**	**Test I**	**Test II**
Granátové jablko	501	32	Nedostupné
Liči	633	11	Nedostupné
Kiwi	935	30	14
Papája	529	10	Nedostupné
Cantaloupe	539	36	Nedostupné
Citron	535	30	10
Pomeranč	861	30	22
Jahoda	1017	30	30
Malina	609	31	12
Borůvka	450	30	4
Jablko	696	30	27

Tabulka 5.1: Počty obrázků ovoce pro jednotlivé části datasetu.

V původní trénovací sadě bylo zahrnuto celkem 7305 obrázků. Mezi nimi byly jak fotografie samostatného ovoce daného druhu a to z různých úhlů i vzdáleností od objektivu fotoaparátu, tak i fotografie většího počtu kusů daného druhu ovoce i různé bedny nebo regály s ovocem, které se podobají způsobu vystavení ovoce v obchodech. Příklady fotografií z datasetu jsou na obrázku 5.4. Počty obrázků v jednotlivých částech datasetu a v datase-

tech pro experimenty shrnuje tabulka 5.1. Validační obrázky, kterých bylo použito celkem 1100, byly vybrány ze složek, obsahujících testovací data. Testovací dataset pro první experiment obsahoval 300 obrázků, sesbíraných z neplacených fotobank[1]. Některé méně obvyklé druhy ovoce měly pro první testovací dataset menší množství obrázků, protože se na jmenovaných fotobankách téměř nevyskytovaly (byly k nalezení pouze na placených fotobankách). Dataset pro druhý experiment měl celkem 119 obrázků.

Po natrénování sítě s využitím datasetu ImageNet bylo dosaženo poměrně slušných výsledků rozpoznávání při testování sítě v rámci prvního experimentu. Následující fázi tvořil druhý experiment, při kterém jsem testovala síť na vlastních obrázcích. Tyto obrázky jsem sesbírala pomocí mobilního telefonu. Obrázky už nebyly všechny tak kvalitní jako ty v datasetu pro první experiment, a to kvůli různým světelným podmínkám, občasné neostrosti apod. Druhy, u kterých nebyl dostatek nafocených obrázků, protože se příliš neprodávají nebo pro ně nebyla sezóna v době sběru dat, mají množství obrázků nižší. Příklady fotografií z tohoto datasetu jsou na obrázku 6.2.

[1] Těmito fotobankami byly www.freeimages.com, www.stockvault.net a www.commons.wikimedia.org.

Kapitola 6

Experimenty

Experimenty byly rozděleny do 2 částí. V první z nich bylo cílem otestovat úspěšnost rozpoznávání vybraných 11 druhů ovoce se sítí, natrénovanou s využitím datasetu ImageNet. Ačkoliv Caffe při rozpoznávání objektů vypisuje 5 nejvíce pravděpodobných klasifikačních tříd spolu s percentuální pravděpodobností pro každou z nich, úspěšné rozpoznání u konkrétního druhu ovoce znamenalo, že byla správná třída mezi prvními třemi položkami z tohoto seznamu. Tyto třídy jsou totiž zasílané uživateli mobilní aplikace.

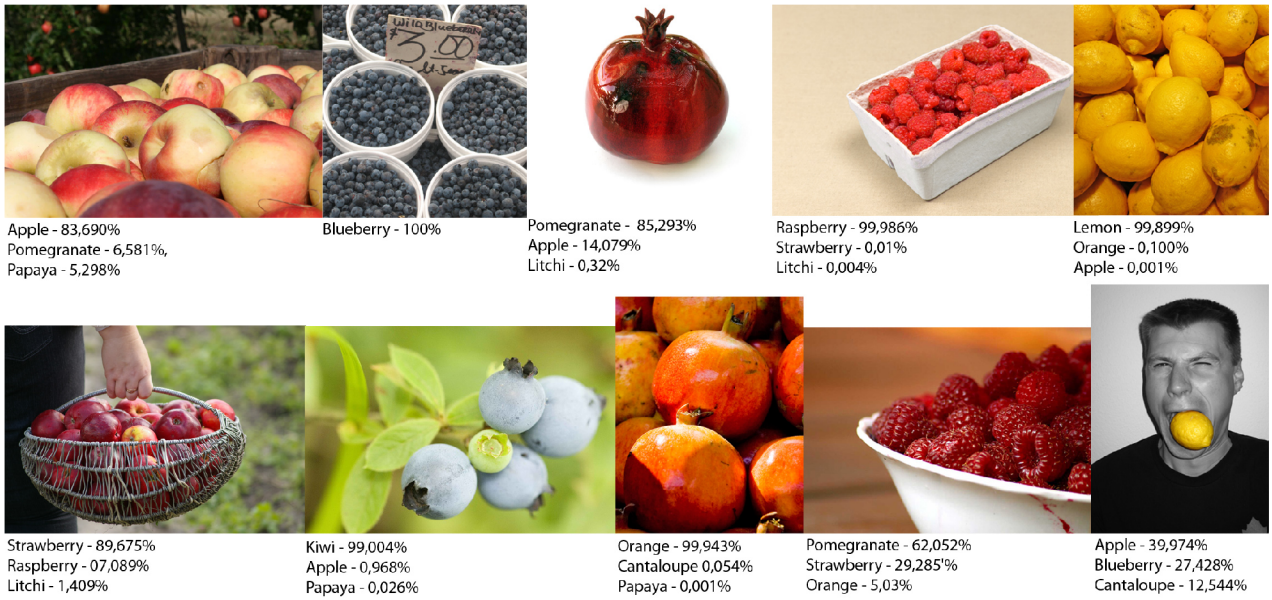
Druhý experiment spočíval v ověření sítě na vlastnoručně vytvořených obrázcích. V další části následuje popis provedených experimentů. Úspěšností pro jednotlivé části datasetu jsou zaznamenány v tabulce 6.1. Podrobné výsledky experimentů byly pro jejich rozsáhlost umístěny na přiložené DVD.

6.1 Experiment s datasetem z fotobank

První experiment spočíval v natrénování sítě s využitím datasetu ImageNet, který je popsán v sekci 5.6, a jejím ověření na testovací sadě dat. Původní dataset byl ještě před tréninkem ručně protříděn pro odstranění nerelevantních obrázků. Dále bylo využito logovacího souboru pro sledování průběhu klasifikace obrázků z testovací sady dat, zaslaných do složky `upload`. Z výsledků lze vyvodit, že pokud jsou síti předloženy kvalitní obrázky z databanky focené za dobrých světelných podmínek a o dostatečném rozlišení, síť má poměrně vysokou úspěšnost rozpoznávání pro zkoumané druhy ovoce.

Na obrázku 6.1 jsou uvedeny příklady fotografií, které byly správně (první řádek) a špatně (druhý řádek) klasifikovány pro 5 vybraných druhů ovoce včetně procent pravděpodobnosti pro první 3 třídy. Z této ukázky byly pouze borůvky s naprostou jistotou klasifikovány do jediné třídy. Je vidět, že klasifikace funguje poměrně přesně pro požadované obrázky, a hůře rozpoznává ty s horšími světelnými podmínkami nebo ty, které obsahují další rušivé prvky jako matoucí pozadí, nezvyklé objekty apod. U borůvek nebyl rozpoznán obrázek s větvičkou, protože těchto obrázků v trénovací sadě nebylo ponecháno mnoho, jelikož neodpovídají zaměření aplikace.

Výsledky experimentu se nachází v tabulce 6.1. Z ní je vidět, že pro první experiment byly dokonce 3 druhy rozpoznány se stoprocentní úspěšností, a u žádného druhu neklesla úspěšnost rozpoznávání pod 80%.



Obrázek 6.1: Příklady klasifikace pro vybrané fotografie.

6.2 Experiment s vlastním datasetem

Druhý experiment spočíval nejdříve v přípravě nového datasetu sesbíráním fotografií v obchodech a doma s využitím aplikace LookAtIt. S touto databází bylo opět provedeno vyhodnocení úspěšnosti rozpoznávání. Jelikož pro některé druhy ovoce v době tvorby tohoto datasetu nebyla sezóna a jiné se příliš neprodávají, byly ověřeny pouze v rámci prvního experimentu. Výsledky experimentu shrnuje tabulka 6.1. Úspěšnost rozpoznávání byla oproti prvnímu experimentu nižší, například u malin, nafocených především v krabičkách v obchodě. V tomto datasetu byly také méně kvalitní nebo i rozmazané snímky.

Obrázek 6.2: Příklady obrázků z vlastního datasetu.

6.3 Závěr z experimentů a možnosti budoucího rozšíření aplikace

Navržené a výše popsané experimenty prokázaly, že pokud uživatel mobilní aplikace vyfotí a předloží neuronové síti kvalitní snímky, je úspěšnost rozpoznávání poměrně vysoká. Pro oba experimenty byla u 3 druhů úspěšnost rozpoznávání dokonce 100%. V rámci prvního experimentu navíc úspěšnost neklesla pod 80% a byla celkově vyšší, než v případě druhého experimentu. To je dáno kvalitou fotografií, jelikož pro první experiment byly použity fotografie z fotobank, kdežto ve druhém byly vytvořeny vlastní. V případě experimentu s vlastními fotografiemi úspěšnost rozpoznávání kromě jednoho druhu neklesla pod 50%. Neuronová síť v aktuálním stavu by tedy mohla být dobře použitelná pro rozpoznávání vybraných druhů ovoce. Pro statisticky významnější výsledky by bylo třeba především dataset druhého experimentu rozšířit o desítky až stovky dalších položek.

Druh	**Úspěšnost E1**	**Úspěšnost E2**
Granátové jablko	90,63%	Nedostupné
Liči	100%	Nedostupné
Kiwi	100%	57,14%
Papája	100%	Nedostupné
Cantaloupe	97,22%	Nedostupné
Citron	86,67%	100%
Pomeranč	93,33%	100%
Jahoda	96,67%	93,33%
Malina	83,87%	25%
Borůvka	96,67%	100%
Jablko	80%	62,96%

Tabulka 6.1: Shrnutí experimentů

Možností dalšího rozšíření vytvořené aplikace je mnoho a jejich výběr by záležel především na způsobu dalšího využití aplikace. Celkově by bylo možné rozšířit dataset o další druhy ovoce, jako jsou například exotické druhy, které se u nás příliš nevyskytují a lidé mohou mít s jejich poznáváním při nákupu problémy. Šlo by také přidat další metodu detekce a rozpoznávání objektů v obraze, aby se zvýšila přesnost klasifikace. Prakticky neustále pak lze rozšiřovat trénovací dataset co do počtu snímků a to například za použití těch, které by uživatelé v rámci zpětné vazby při špatném rozpoznání druhu vraceli na server s určenou správnou klasifikační třídou. Dále by také bylo možné, aby po zpracování obrázku byl tento přemístěn do složky `processed`, a na základě úspěšnosti klasifikace by se s ním mohly provádět další kroky (např. použít jej k dalšímu učení sítě). Aplikaci by také šlo rozšířit ve směru využití pro zrakově postižené, aby jim pomáhala jako rádce při nákupu.

Kapitola 7

Závěr

Cílem této práce bylo navrhnout a implementovat především serverovou část aplikace s klient-server architekturou, využitelnou pro rozpoznávání objektů v obraze. Tento záměr se podařilo splnit. Nejdříve jsem se seznámila s dostupnou literaturou a softwarem se zaměřením na detekci a rozpoznání objektů v obraze a na tvorbu mobilních klient-server aplikací, přičemž jsem na začátku uvažovala o detekci obličejů nebo aut. Po dalším přemýšlení jsem ale jako objekt zájmu zvolila druhy ovoce, protože takováto aplikace by se mohla hodit například při nákupu zboží v obchodě nebo po vhodné úpravě jako pomocník nevidomých při nákupu. Jako metodu detekce a rozpoznání objektů v obraze jsem zvolila umělé neuronové sítě, které mě zaujaly svými vlastnostmi, díky nimž jsou vhodné pro rozpoznávání objektů v obraze. Z množství frameworků a knihoven, které pracují s neuronovými sítěmi, jsem k práci zvolila framework Caffe, protože má kvalitní dokumentaci a dosahuje vysokých rychlostí při zpracování obrazu.

Výsledkem práce je mobilní klient-server aplikace, zahrnující na straně serveru umělou neuronovou síť, vytvořenou za použití frameworku Caffe, dále rozhraní serveru a také existující mobilní aplikaci LookAtIt, přizpůsobenou pro účely práce. Následovalo ověření funkčnosti aplikace pomocí experimentů, jež prokázaly, že pokud jsou síti předloženy kvalitní fotografie, je rozpoznávací schopnost poměrně vysoká. Funkčnost aplikace byla během experimentů demonstrována na dvou sadách dat, které jsou k nalezení na přiloženém CD.

V průběhu prvního experimentu s datasetem z fotobank bylo dosaženo u 3 druhů dokonce stoprocentní úspěšnosti rozpoznávání. U ostatních druhů neklesla úspěšnost rozpoznávání pod 80%. Při druhém experimentu s vlastními fotografiemi bylo dosaženo stoprocentní úspěšnosti rozpoznávání také u 3 tříd a u ostatních zkoumaných tříd kromě malin se úspěšnost rozpoznávání pohybovala nad 50%. Z toho je vidět, že je tato neuronová síť dobře prakticky použitelná pro rozpoznávání 11 vybraných druhů ovoce, a to i pro vlastní fotografie uživatele, které mohou být neostré nebo méně kvalitní podobně jako některé fotografie z datasetu druhého experimentu.

Tato bakalářská práce mi umožnila udělat si hlubší představu o oborech detekce a rozpoznání objektů v obraze a získala jsem nové vědomosti z oblasti neuronových sítí. Vytvoření původního návrhu součástí serverové části se ukázalo být obtížnější, než jsem očekávala, nakonec jsem si s ním ale poradila a navržené řešení funguje efektivně, jak prokázaly experimenty v sekci 6.

V práci bych ráda dále pokračovala přidáním další rozpoznávací metody (například jednoho z boosting algoritmů, zmíněných v sekci 3), která by v kombinaci s neuronovou sítí vedla k ještě vyšší úspěšnosti rozpoznávání objektů v obraze. Také by bylo zajímavé rozpoznávat exotické ovoce, které u nás není příliš běžné. V práci bych také jistě mohla

pokračovat směrem vytvoření aplikace pro nevidomé, ale toto zaměření by ještě vyžadovalo získání znalostí, týkajících se vývoje aplikací pro nevidomé.

Literatura

[1] ALOIMONOS, Y.; ROSENFELD, A.: Computer Vision. *Science*, ročník 253, č. 5025, 1991.

[2] AYACHE, N.: *Computer Vision, Virtual Reality and Robotics in Medicine: First International Conference, CVRMed '95, Nice, France, April 3 - 6, 1995. Proceedings.* Springer, 1995, ISBN 978-3-540-59120-7, 567 s.

[3] BUREŠ, L.; ZIMMERMANN, P.: Metody počítačového vidění: 9. přednáška: Strojové učení. http://www.kky.zcu.cz/uploads/courses/mpv/09/lesson09_materialy.pdf, 2014, [cit. 2015-01-08].

[4] Caffe.berkeleyvision.org: Blobs, Layers and Nets: anatomy of a Caffe model. http://caffe.berkeleyvision.org/tutorial/net_layer_blob.html, 2015, [cit. 2015-02-03].

[5] Caffe.berkeleyvision.org: Caffe Model Zoo. http://caffe.berkeleyvision.org/model_zoo.html, 2015, [cit. 2015-05-02].

[6] Caffe.berkeleyvision.org: Solver. http://caffe.berkeleyvision.org/tutorial/solver.html, 2015, [cit. 2015-03-03].

[7] Camfindapp.com: CamFind [online]. http://camfindapp.com/, 2015, [cit. 2015-04-05].

[8] CHALUPNÍK, V.: Biologické algoritmy (4) – Neuronové sítě. http://www.root.cz/clanky/biologicke-algoritmy-4-neuronove-site/, 2012, [cit. 2015-02-02].

[9] Cyber.felk.cvut.cz: Neuronové sítě: Úvod do neuronových sítí. http://cyber.felk.cvut.cz/gerstner/biolab/bio_web/teach/FunBio/neuron/neursite.html, 2015, [cit. 2015-01-06].

[10] Deeplearning4j.org: DL4J [online]. http://deeplearning4j.org/, 2015, [cit. 2015-02-06].

[11] DRAHANSKÝ, M.; ORSÁG, F.; kolektiv: *Biometrie.* Computer Press, 2011, ISBN 978-80-254-8979-6, 294 s.

[12] ELLEITHY, K.; SOBH, T.: *New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering.* Springer, 2014, ISBN 978-3-319-06763-6, 635 s.

[13] GAGALOWICZ, A.; PHILIPS, W.: 5th International Conference, MIRAGE 2011, Rocquencourt, France, October 10-11, 2011. Proceedings. In *Computer Vision/Computer Graphics Collaboration Techniques*, 2011.

[14] HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, druhé vydání, 2009, ISBN 978-0-387-84857-0, 745 s.

[15] HLAVÁČ, V.: Digitální zpracování obrazu, počítačové vidění zakotvení: přednáška [online]. http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/012IntroDigImProcVsCompVisionCz.pdf, [cit. 2015-01-02].

[16] HLAVÁČ, V.; ŠONKA, M.: *Počítačové vidění*. GRADA, 1992, ISBN 80-85424-67-3.

[17] HORZYK, A.: Artificial Neural Networks. http://home.agh.edu.pl/~horzyk/lectures/ai/ann.php, 2015, [cit. 2015-04-08].

[18] IONESCU, B.; BENOIS-PINEAU, J.; PIATRIK, T.; aj.: *Fusion in Computer Vision: Understanding Complex Visual Content*. Springer, 2014, ISBN 978-3-319-05695-1, 286 s.

[19] JAIN, A. K.; MAO, J.; MOHIUDDIN, K.: Artificial Neural Networks: A Tutorial [online]. *Computer*, ročník 29, č. 3, 1996.

[20] JAKLIN, P.: Neuronové sítě. http://cgg.mff.cuni.cz/~pepca/prg022/mucha/, 2004, [cit. 2015-04-01].

[21] JIA, Y.; SHELHAMER, E.; DONAHUE, J.; aj.: Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[22] JURÁNEK, R.; ZEMČÍK, P.; HRADIŠ, M.: Real-Time Algorithms of Object Detection Using Classifiers [online]. http://cdn.intechopen.com/pdfs-wm/35324.pdf, 2012, [cit. 2015-04-09].

[23] KALOVÁ, I.: Předzpracování obrazu: přednáška [online]. http://midas.uamt.feec.vutbr.cz/POV/lectures-pdf/02_Predzpracovani_obrazu.pdf, 2013, [cit. 2015-05-7].

[24] KARPATHY, A.: Convolutional Neural Networks for Visual Recognition [online]. http://cs231n.github.io/convolutional-networks/, 2015, [cit. 2015-02-14].

[25] KAČENKA, P.: Neuronové sítě [online]. http://mks.mff.cuni.cz/library/NeuronoveSitePK/NeuronoveSitePK.pdf, 1998, [cit. 2015-02-02].

[26] Layar.com: Layar [online]. https://www.layar.com/, 2015, [cit. 2015-04-05].

[27] Linux.die.net: Incrond(8) – Linux man page [online]. http://linux.die.net/man/8/incrond, 2015, [cit. 2014-12-02].

[28] Looktel.com: LookTel [online]. http://www.looktel.com/, 2015, [cit. 2015-04-05].

[29] Láník, A.; kolektiv: LookAtIt pro Android [online]. https://play.google.com/store/apps/details?id=com.lookatit.android, [cit. 2015-05-04].

[30] MALTAROLLO, V. G.; SILVA, K. M. H. A. B. F. D.: *Artificial Neural Networks - Architectures and Applications*. InTech, 2013, ISBN 978-3-540-59120-7, 264 s.

[31] Mathworks.com: Image Processing Toolbox [online]. http://www.mathworks.com/products/image/index-b.html, 2015, [cit. 2015-02-08].

[32] MCCULLOCH, W. S.; PITTS, W.: A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, ročník 5, č. 4, 1943. URL http://dx.doi.org/10.1007/BF02478259

[33] MICHALSKI, R. S.; TECUCI, G.; CARBONELL, J. G.; aj.: *Machine Learning: An Artificial Intelligence Approach*. M. Kaufmann, Čtvrté vydání, 1994, ISBN 978-1-55860-251-9, 782 s.

[34] MICHIE, D.; SPIEGELHALTER, D. J.; TAYLOR, C.: Machine Learning, Neural and Statistical Classification [online]. http://www1.maths.leeds.ac.uk/~charles/statlog/whole.pdf, 1994, [cit. 2015-03-05].

[35] OLAH, C.: Conv Nets: A Modular Perspective [online]. http://colah.github.io/posts/2014-07-Conv-Nets-Modular/, 2014, [cit. 2015-01-04].

[36] OpenCV.org: OpenCV [online]. http://opencv.org/, 2015, [cit. 2015-02-04].

[37] OpenDeep.org: OpenDeep [online]. http://www.opendeep.org/, 2015, [cit. 2015-02-03].

[38] Otexts.org: Boosting [online]. https://www.otexts.org/1538, 2015, [cit. 2015-04-01].

[39] ROJAS, R.: *Neural Networks: A Systematic Introduction*. Springer-Verlag, 1996, ISBN 3-540-60505-3, 803 s.

[40] RUSSAKOVSKY, O.; DENG, J.; SU, H.; aj.: ImageNet Large Scale Visual Recognition Challenge [online]. 2014. URL http://www.image-net.org/

[41] SHIFFMAN, D.: Chapter 10. Neural Networks [online]. http://natureofcode.com/book/chapter-10-neural-networks/, 2012, [cit. 2015-04-06].

[42] Solver.com: Training an Artificial Neural Network: Intro [online]. http://www.solver.com/training-artificial-neural-network-intro, 2015, [cit. 2014-12-16].

[43] Torch.ch: Torch [online]. http://torch.ch/, 2015, [cit. 2015-02-06].

[44] TRENZ, O.; FEJFAR, J.; POPELKA, O.; aj.: Umělá inteligence: eLearningová opora [online]. http://is.mendelu.cz/eknihovna/opory/index.pl?cast=21471, 2010, [cit. 2015-02-14].

[45] VASILEV, I.: A Deep Learning Tutorial: From Perceptrons to Deep Networks. http://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks, 2014, [cit. 2015-04-01].

[46] VIOLA, P.; JONES, M.: Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001.

[47] VOLNÁ, E.: Neuronové sítě 1: Skripta [online]. http://www1.osu.cz/~volna/Neuronove_site_skripta.pdf, 2008, [cit. 2014-10-18].

[48] YANG, M.-H.: Object Recognition [online]. http://faculty.ucmerced.edu/mhyang/papers/object-recognition-chapter.pdf, 2009, [cit. 2015-03-13].

[49] YANG, M.-H.; AHUJA, N.: *Face Detection and Gesture Recognition for Human-Computer Interaction*. Springer US, 2001, ISBN 978-1-4613-5546-5, 182 s.

[50] ŠIKL, R.: *Zrakové vnímání*. GRADA, 2012, ISBN 978-80-247-3029-5, 312 s.

[51] ČERMÁK, J.: *Implementace neuronové sítě do mikrokontroléru*. Diplomová práce, Brno: Vysoké učení technické. Fakulta elektrotechniky a komunikačních technologií, 2009.

Dodatek A

Obsah DVD

- Elektronická verze bakalářské práce
- Neuronová síť s potřebnými soubory
- Uživatelský návod k programu
- Databáze snímků k experimentům
- Kompletní výsledky experimentů ve formátu PDF

Dodatek B

Postup instalace frameworku Caffe

Caffe framework zahrnuje rozhraní pro Python (`pycaffe`) i pro MATLAB. Framework lze nainstalovat a používat na systémech Ubuntu 14.04 a 12.04, MAC OS X 10.10/10.9/10.8, AWS a na dalších linuxových distribucích, kterými jsou RHEL, Fedora a CentOS. Instalace vyžaduje několik prerekvizit:

- knihovnu CUDA pro použití Caffe v GPU režimu (nejlépe ve verzi 7.0, popř. verze 5.0, 5.5 a 6). Caffe vyžaduje CUDA `nvcc` kompilátor pro kompilaci GPU kódu, a také CUDA ovladač pro GPU. Caffe lze akcelerovat pomocí NVIDIA cuDNN.
- BLAS knihovnu, nutnou pro back-end pro maticové a vektorové výpočty. Lze vybrat z různých implementací této knihovny – ATLAS, Intel MKL či OpenBLAS.
- C++ knihovny `Boost` verze 1.55 nebo vyšší.
- OpenCV verze 2.4 včetně podpory verze 3.0.
- `protobuf, glog, gflags`.
- knihovny `hdf5, leveldb, snappy, lmdb`.

Pro Pycaffe (Python Caffe) je třeba Python verze 2.7 nebo 3.3+, `numpy` (ver. 1.7 a vyšší), `boost.python`. Pro MATLAB Caffe je potřebný MATLAB spolu s `mex` kompilátorem.

B.1 Kompilace Caffe frameworku

Po instalaci a zprovoznění všech prerekvizit lze kompilovat Caffe. Je nutné upravit soubor `Makefile.config` tak, aby obsahoval korektní cesty. Původní nastavení by mělo fungovat, ale při použití Anaconda Pythonu je třeba odkomentovat relevantní řádky v souboru. Pro cuDNN akceleraci je nutné v souboru `Makefile.config` odkomentovat `USE_CUDNN:=1` přepínač. Při běhu Caffe na CPU stačí odkomentovat řádek `CPU_ONLY:=1` ve stejném souboru.

Pro kompilaci wrapperů pro Python a MATLAB je třeba použít příkazů `make pycaffe`, respektive `make matcaffe`. Je nutné mít nejdříve nastavené MATLAB a Python cesty v souboru `Makefile.config`. Pro distribuované výpočty je třeba použít příkaz `make distribute`, který vytvoří adresář s názvem `distribute` se všemi Caffe hlavičkami, zkompilovanými knihovnami, binárními a dalšími soubory, které jsou potřebné pro distribuci na jiná zařízení. Pro rychlejší sestavení je možné použít paralelní kompilace pomocí příkazu `make all -j8`, kde 8 je počet paralelních vláken pro kompilaci (například počet jader procesoru).