VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A ROZPOZNÁNÍ HUB V PŘIROZENÉM PROSTŘEDÍ

MUSHROOM DETECTION AND RECOGNITION IN NATURAL ENVIRONMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. DOMINIK STEINHAUSER**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAKUB ŠPAŇHEL**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Steinhauser Dominik, Bc.**
Obor: Matematické metody v informačních technologiích
Téma: **Detekce a rozpoznání hub v přirozeném prostředí**
Mushroom Detection and Recognition in Natural Environment
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na algoritmy detekce a rozpoznávání.
2. Vyberte vhodné metody a navrhněte řešení problému detekce a rozpoznání hub v přirozeném prostředí.
3. Experimentujte s vaší implementací a případně navrhněte vlastní modifikace metod.
4. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
5. Vytvořte stručný plakát a video prezentující vaši bakalářskou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních tří bodů zadaní.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Špaňhel Jakub, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Práce řeší problém detekce a klasifikace hub na fotografiích v přirozeném prostředí. K řešení jsem použil konvoluční neuronové sítě. Začátek práce je věnován teorii neuronových sítí. Dále je v práci řešena detekce hub a jejich klasifikace. S pomocí plně konvoluční neuronové sítě je vyřešen i problém lokalizace hub. Výsledky naučených neuronových sítí jsou analyzovány.

Abstract

In this thesis is handled the problem of mushroom detection and recognition in natural environment. Convolutional neural networks are used. The beginning of this thesis is dedicated to the theory of neural networks. Further is solved the problem of object detection and classification. Using neural network trained for classification is solved also the task of localization. Results of trained CNNs are analised.

Klíčová slova

Strojové učení, Konvoluční neuronové sítě, Klasifikace, Klasifikace hub, Detekce hub, CNN, FCN

Keywords

Machine learning, Convolutional neural networks, Classification, Mushroom classification, Mushroom detection,CNN, FCN

Citace

STEINHAUSER, Dominik. *Detekce a rozpoznání hub v přirozeném prostředí.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jakub Špaňhel

Detekce a rozpoznání hub v přirozeném prostředí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Špaňhela. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Dominik Steinhauser
24. května 2017

Poděkování

Děkuji Ing. Špaňhelovi za obětavé vedení mé práce.

Obsah

Kapitola 1

Úvod

V České republice i v jiných zemích je velmi populární koníček houbaření. Většina houbařů však dokáže určit pouze malou část u nás rostoucích druhů hub. Zkušený houbař rozeznává okolo padesáti druhů hub. Mykolog až přibližně tisíc druhů[1]. Existují atlasy hub ať už ve formě papírových knih nebo ve formě aplikací, ve kterých je možné vyhledávat podle jména a získat popis houby, rozlišující znaky a fotografie. Možnosti jejich použití jsou však velmi omezené, protože pokud houbař houbu nezná, nemůže ji v atlasu vyhledat. Rozhodl jsem se vytvořit program, který bude houbu klasifikovat na základě fotografie. Výstupem programu budou návrhy o jaký druh se může jednat[2]. Houbař potom už může použít klasický atlas, prostudovat si popis houby a rozhodnout, zda se skutečně jedná o daný druh. Experimentoval jsem s různými přístupy, jak tento problém řešit a ukázalo se, že nejúspěšnější jsou konvoluční neuronové sítě(CNN). Pomocí CNN byl v prostředí Caffe implementována aplikace umožňující detekovat a klasifikovat houby. Výsledky jsou použitelné v praxi a budou využity v aplikaci pro platformu android.

Práce je dělena do níže popsaných kapitol. Po tomto úvodu jsou v kapitole 2 popsány existující řešení klasifikace hub a podobných problémů.

Kapitola 3 pojednává o fungování a učení neuronových sítí především o CNN, které jsou zvláště vhodné pro klasifikaci obrazu.

V kapitole 4 jsou popsány algoritmy použité pro lokalizaci objektů, především plně konvoluční neuronové sítě (FCN), které se ukázaly být nejúspěšnější.

Kapitola 5 popisuje experimenty s metodami pro rozpoznání hub a použité metody klasifikace především použité CNN.

V kapitole 6 je představena vytvořená datová sada a strukturu dat, se kterými tato práce pracuje.

Dále 7 jsou popsány použité nástroje, především framework Caffe, který umožňuje návrh CNN a jejich učení pomocí grafické karty.

V kapitolách 8 a 9 jsou popsány dosažené výsledky, shrnuta vykonaná práce a představeny možnosti použití získaných výsledků a možnosti, jak danou problematiku dále rozvinout.

[1] Jedná se o subjektivní odhad několika mykologů.

[2] V ideálním případě bude návrh samozřejmě pouze jeden.

Kapitola 2

Existující řešení

Na rozdíl od detekce a klasifikace lidských obličejů, postav, zvířat atd. je problém klasifikace hub téměř nezpracovaný. Proto nebylo možné stáhnout si existující datovou sadu, ale bylo potřeba vytvořit si vlastní, která bude nejlépe odpovídat řešenému problému.

Problém klasifikace hub pomocí počítače byl zatím řešen pouze okrajově. Problém byl řešen například na Kalifornské univerzitě [16]. Je zde použita datová sada [27] obsahující 8042 instancí 23 druhů hub, patřící do jediného rodu pečárkovité. Jednotlivým exemplářům jsou přiřazeny hodnoty 22 atributů jako jsou například vůně, tvar klobouku, barva apod. Každý atribut nabývá pouze několika, nejvýše 10 hodnot. Pro klasifikaci je použit algoritmus k-nejbližších sousedů [1] a naivní Bayesův klasifikátor. Pomocí této techniky je dosažena úspěšnost 87 procent respektive 93 pro rozlišení, zda se jedná o jedlou či jedovatou houbu. Úspěšnost klasifikace na jednotlivé druhy není uvedena. Daný přístup není v praxi použitelný hned z několika důvodů. Za prvé se klasifikuje pouze v rámci jednoho rodu a není nijak ověřeno, že dosud nezařazená houba do tohoto rodu skutečně patří. Dále je zde problém, že zadávání jednotlivých příznaků často není jednoznačné a snaze o úplné vymezení všech možností zase neuměrně narůstá jejich počet.

Úspěšnějším příkladem je interaktivní klasifikátor MycoKey [19]. S výše uvedeným řešení mají společné to, že klasifikace neprobíhá automaticky z fotografie, nýbrž vyžaduje ruční zadání atributů jako jsou barva, velikost, vůně, povrch, místo nálezu. Atributů jsou zde desítky. Z těchto se provede klasifikace. Ve volně dostupné verzi je klasifikováno 600 druhů hub. Po zakoupení placené verze se jedná až o 3300 druhů hub. Interaktivní použití probíhá tak, že se po zadání několika atributů aktualizuje seznam druhů, které přichází v úvahu. Aplikace průběžně vyhodnocuje, jaké atributy nejlépe rozlišují vybrané druhy a vyžaduje jejich vyplnění. V okamžiku, kdy je určen druh houby, je zobrazen jeho detailní popis a příklady fotografií, což umožňuje zpětnou kontrolu. Autor neuvádí úspěšnost ani použité metody klasifikace, možnost zpětné kontroly však riziko omylu minimalizuje. Při pokusném zadání 25 náhodných exemplářů hub došlo ke správnému zařazení ve všech případech. Průměrná doba klasifikace však byla 1:46 minut, což je pro běžné použití poměrně dlouhý čas. Slabinou tohoto řešení je opět nejednoznačnost vybíraných atributů, která dělá ze zadávání poměrně náročný úkol[1].

Na naší univerzitě vznikla bakalářská práce, která se klasifikaci věnovala [48]. Autor se pokusil vyvinou vlastní metodu. Z obrázku extrahoval 6 typů příznaků a poté houby s jejich pomocí klasifikoval. Klasifikoval ale pouze 6 různých druhů s úspěšností 72%, což je pro praktické využití nedostatečné. Omezen byl i velikostí datové sady, která obsahovala

[1] Už například výběr nejpodobnější barvy klobouku z 50 možností není otázka několika sekund

pouze několik stovek obrázků, protože autor používal pouze ručně vybrané fotografie, kde je houba vyfotografována pod ideálním úhlem.

2.1 Klasifikace květin Oxford

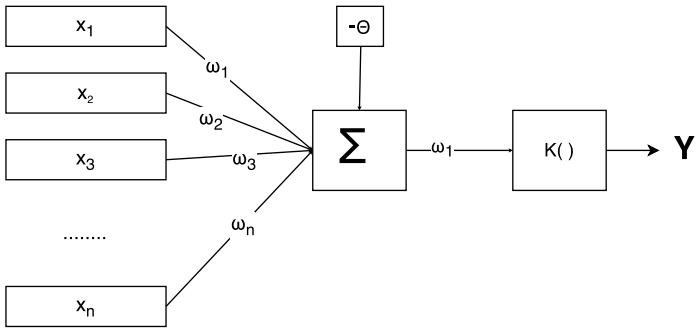
Tématu mé práce se nejvíce podobá řešení klasifikace 102 druhů květin v Oxfordské datové sadě. Nejprve byla pomocí barvy vyřešena segmentace [32]. Dále autor použil kombinaci příznaků, popisující barvu, texturu a tvar. Jejich kombinací dosáhl top-1 úspěšnost 87%.Řešení bylo dále vylepšeno použitím konvolučních neuronových sítí, kde autor dosáhl top-1 úspěšnosti 93% při použití sítě AlexNet.

V této práci použiji podobný postup. Náročnějším úkolem se v mé datové sadě ukazuje lokalizace objektu, protože fotografií i kategorií je řádově větší počet a není proto možné objekty ručně lokalizovat.

Kapitola 3

Konvoluční neuronové sítě

Ke klasifikaci obrázků do kategorií se používají různé metody strojového učení. Po úspěchu Alexe Krizhevsky [4], na soutěži NIPS, se velmi zpopularizovalo používání konvolučních neuronových sítí. Neuronové sítě jsou výpočetní model inspirovaný způsobem zpracování informací v lidském mozku. Poprvé byly popsány v roce 1951, ale více se začaly využívat až od roku 1974, kdy byl popsán algoritmus zpětného šíření chyby[1]. Výpočet je paralelizován do jednotlivých neuronů, které spolu komunikují. Použití neuronových sítí v poslední době umožnila mimo jiné dostupnost výkonných počítačů, které umožňují trénování velkých sítí v rozumném čase.

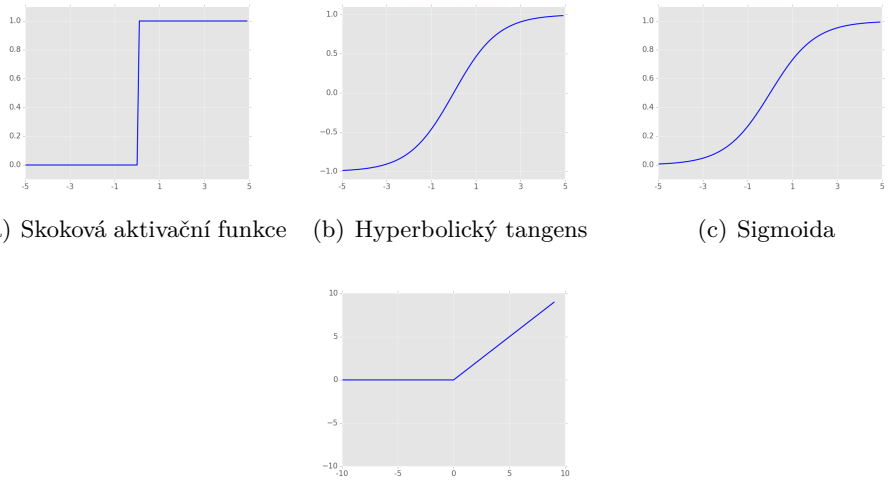


Obrázek 3.1: Obrázek ukazuje schéma neutronu. Použito je následující značení: x_i - vstupy neuronu, ω_i - váhy synapsí, Θ - práh, $K()$ - aktivační funkce neuronu, Y - výstup

3.1 Definice neuronu

Neuron je základní jednotka výpočtu v neuronových sítích. Je inspirován biologií. Výpočet v neuronu probíhá podle schématu znázorněného na obrázku 3.1. Vstupy jsou násobeny vahami synapsí a od jejich součtu je odečten aktivační práh neuronu[2]. Na výsledek je poté aplikována nelineární aktivační funkce. Aktivační funkce je diferencovatelná, což je, jak později ukážeme důležité pro učení sítě. V modelu model popsaném McCullochem a Pittsem se jako aktivační funkce používá skoková funkce [30]. Ve frameworku Caffe, který jsem pro práci s neuronovými sítěmi používal se lze setkat i se sigmoidální funkcí, funkcí hyperbolické tangenty a dalšími [20]. Nejčastěji se však používá lomená lineární funkce s prahem 0. Grafy zmíněných aktivačích funkcí jsou na obrázku 3.2.

[1] anglicky backpropagation
[2] anglicky bias



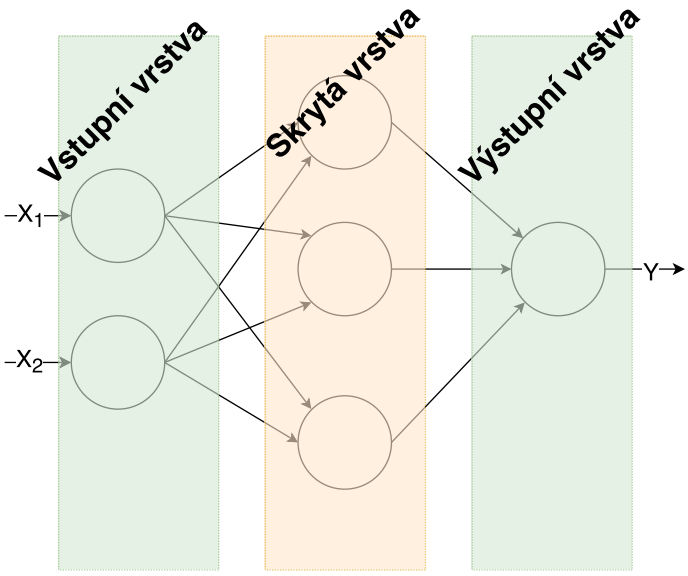
(a) Skoková aktivační funkce (b) Hyperbolický tangens (c) Sigmoida

(d) Relu

Obrázek 3.2: Příklady aktivačních funkcí

3.2 Struktura sítě

Neurony jsou uspořádány do několika vrstev. Pokud je vrstev více než 2, nazývá se hluboká neuronová síť. Vstupy sítě jsou přijímány vstupní vrstvou. Dále síť obsahuje několik skrytých vrstev a výstupy poslední vrstvy jsou považovány za výstup sítě. Příklad sítě s jednou vstupní skrytou i výstupní funkcí je na obrázku 3.3.



Obrázek 3.3: Příklad struktury sítě se třemi vrstvami. Síť má dva vstupy x_1, x_2 a jeden výstup y. Šipky znázorňují propojení neuronů.

3.2.1 Výpočet výstupu sítě

Výpočet probíhá tak, že jsou data přivedena do vstupní vrstvy. Každý neuron v ní obsažený vypočte výsledek své aktivační funkce a tento výsledek je převeden na vstup všech neuronů v další vrstvě, se kterými je tento neuron spojen. V praxi se tento výpočet provádí jako násobení matic a aplikace nelineárních funkcí zastupujících aktivační funkci. Tento postup můžeme ilustrovat na příkladu sítě na obrázku 3.3. Jedná se o síť se dvěma vstupy, jednou skrytou vrstvou a jednou výstupní vrstvou. Předpokládejme, že aktivační funkce všech neuronů je skoková s prahem 0 a na vstup přichází vektor (1,-1), potom aplikací vzorce pro výpočet výstupu perceptronu dostáváme:

$$Y_1 = S((\sum_{i=0}^{n} \omega_{x_i,y_1})) = S(0) = 0,$$

$$Y_2 = S((\sum_{i=0}^{n} \omega_{x_i,y_2})) = S(1) = 1,$$

$$Y_3 = S((\sum_{i=0}^{n} \omega_{x_i,y_3})) = S(-1) = 0,$$

kde ω_{x_i,y_3} označuje váhu synapse mezi neurony x_i a y_3. a S je skoková funkce. Výpočet lze úsporněji vyjádřit jako součin vektoru vstupů a matice vah synapsí.

$$Y = S(\omega_{x,y} \times X)$$

$$Y = S\left(\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = S\left(\begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{3.1}$$

3.2.2 Trénování sítě

Potom lze výpočet výstupu lze chápat jako funkci, která každému vektoru vstupů přiřadí určitý vektor výstupů. Výsledek je určen jednak strukturou sítě, která je navržena programátorem jednak hodnotami vah synapsí, které lze chápat jako parametry tohoto výpočtu. Pro malé sítě řešící jednoduché problémy by bylo možné tyto váhy nastavit manuálně. Pro řešení náročnějších problémů se však používají rozsáhlé sítě, kdy je tento postup nemožný. K nastavení vah se proto používá algoritmus strojového učení, kdy se váhy nastaví na základě známých vstupů tak, aby potom byli schopné vrátit požadovaný výsledek i na neznámé vstupy. Využíváme-li síť například ke klasifikaci může být výstup poslední vrstvy interpretován jako pravděpodobnost přiřazená jednotlivým kategoriím. Trénováním sítě[3] se potom rozumí hledání takových vah synapsí, aby síť přiřazovala co největší pravděpodobnosti kategoriím, do kterých je zařazen zpracovávaný vstup. Nejpoužívanějším postupem je metoda zpětného šíření chyby. V praxi se tento výpočet realizuje pomocí algoritmu Stochastický gradient descent[4], kdy se váhy mění iterativně, ne na základě celé trénovací sady ale pouze s ohledem na několik trénovacích příkladů [31].

[3] Trénování a učení sítě jsou totožně pojmy.
[4] Stochastic Gradient Descend

Objektivní funkce

Abychom mohli nalézt váhy synapsí takové, které vedou k optimálním výstupům sítě, musíme být schopni určit, jak blízko je výstup neuronové sítě požadovanému výstupu. Toho se dosáhne vyhodnocením tzv. objektivní funkce. Její výpočet probíhá tak, že se porovná předpověď sítě pro daný obrázek s očekávaným výsledkem. Porovnání se provádí metodou střední kvadratické chyby podle následujícího vzorce:

$$f(\omega, b) = \frac{1}{2n} \sum_{x} (y(x) - a)^2 \tag{3.2}$$

Proměnná b zde označuje biasy, ω váhy,x vstupy sítě $y(x)$ výstupy sítě a a očekávané vstupy. Proměnné b ani ω se nevyskytují v pravé straně vzorce, protože jsou obsaženy ve výpočtu funkce $y(x)$, jak bylo popsáno v kapitole 3.2.1.

Čím je objektivní funkce nižší, tím je výsledek blíže požadovanému řešení. Pokud tedy předpověď přesně odpovídá očekávané předpovědi je objektivní funkce nulová.

Algoritmus zpětného šíření chyby

Cílem trénování je tuto objektivní funkci minimalizovat, aby pak síť vracela výsledky, které budou co nejvíce podobné očekávaným výsledkům. Při konstantních vstupech x a očekávaných výstupech a, které jsou v datové sadě je objektivní funkce závislá pouze na hodnotách synapsí a prahů. Minimalizace objektivní funkce proto můžeme realizovat jako hledání minima funkce mnoha proměnných. Jsou li aktivační funkce neuronů navíc spojité funkce, pak je i objektivní funkce spojitá, tedy malá změna některého z parametrů bude mít za následek malou změnu výsledku.

Následující postup lze intuitivně shrnout tak, že při hledání minima se začne kdekoliv, a potom je třeba jít stále z kopce[22], tedy proti směru gradientu. Při hledání minima funkce jedné proměnné by tomuto postupu odpovídal výpočet derivace a odečtení jejího násobku s prahem.

Mějme nějaké hodnoty vah synapsí. Na začátku výpočtu tyto hodnoty vhodně odhadneme. Můžeme je nastavit zcela náhodně nebo mohou být použity hodnoty, sítě se stejnou strukturou, která řeší jiný problém, jak bude detailněji popsáno v podkapitole(3.2.2). Poté budeme hodnoty iterativně měnit. Naším cílem bude najít jiný vektor, který bude vracet nižší hodnotu objektivní funkce. To se provede derivací objektivní funkce přes všechny hodnoty synapsí, čímž se získá gradient, tedy směr tečny objektové funkce. To je v prostoru vah synapsí směr[5], ve kterém objektivní funkce nejvíce roste. Gradient vynásobíme vhodnou nízkou hodnotou δ, která představuje velikost kroku a odečteme od aktuálního vektoru hodnot vah. V jednorozměrném prostoru by tato metoda odpovídala Newtonově metodě hledání minima[7]

Pokud je hodnota velikosti kroku dostatečně malá, je dokázáno [22], že řešení bude konvergovat[6]. Konvergence zaručuje nalezení lokálního nikoliv globálního minima. Čím je však hodnota kroku menší, tím je konvergence pomalejší, což je vzhledem k vysoké výpočetní náročnosti zásadní. Při vyšších hodnotách kroku se naopak zvyšuje riziko, že dojde k divergenci. V praxi se tento problém řeší postupným zmenšováním kroku. Cílem je, aby se na začátku řešení, kdy předpokládáme, že jsme od řešení daleko, dělali velké skoky. Na

[5] Vektor, jehož délka odpovídá počtu vah synapsí

[6] Jedná se o teoretický výsledek platný při výpočtu gradientu ze všech obrázků zároveň, což jak uvidíme později není technicky možné.

konci, kdy už jsme blízko minima, kroky zmenšíme, aby byl výpočet co nejpřesnější. Derivace se provádí podle algoritmu zpětného šíření chyby[7], aby se výpočty neopakovali pro každou synapsi zvlášť. Váhy jednotlivých synapsí se pak pohnou o určenou hodnotu tzv. krok opačným směrem než je směr gradientu.

Při hledání minima funkce je možným problémem uvíznutí v lokálním minimu. Učení sítě je nedeterministické, protože na začátku jsou váhy nastaveny náhodně. Při opakovaném učení se stejnou datovou sadou můžeme tedy skončit pokaždé v jiném lokálním minimu. Praxe však ukazuje, že rozdíl v přesnosti odhadů v těchto lokálních minimech bývá zanedbatelný [31].

V této kapitole byl popsán postup, jak nalézt takové váhy synapsí, aby byla ztrátová funkce na trénovacích datech minimální. Lze očekávat, že budeme čím bude datová sada rozsáhlejší, tím lépe budou tyto váhy dávat přesnější výsledky i na vstupech, které nejsou součástí datové sady. Proces učení je ale omezený výpočetním výkonem a velikostí datové sady, takže váhy, které vrací nejpřesnější výsledky na trénovací sadě nemusí být ideální i na testovacích vstupech. Tomuto jevu se říká přeučení a bude vysvětlen v další kapitole.

Technické aspekty učení

V dosud popsaném postupu jsem dal přednost teoretickému pohledu a dovolil si určité zjednodušení. Představený výklad je totiž pro komplexnější problémy neúnosně výpočetně náročný. Například v síti AlexNet [4], kterou jsem nejvíce používal se vyskytuje 650 000 neuronů v 8 vrstvách, které celkem zpracovávají 500 000 obrázků. K nalezení ustáleného řešení[8] bylo potřeba provést 500 000 iterací. Pokud bychom chtěli v každé iteraci přesně spočítat gradient, museli bychom spočítat výstup sítě a parciální derivace pro pro všechny obrázky. V praxi se proto přesný výpočet gradientu neprovádí. Gradient celé trénovací sady se místo toho aproximuje gradientem pro určitou skupinu tzv dávku[9] obrázků. V praxi se používají dávky ve velikosti od 1 do 512 obrázků[10]. Čím je dávka větší, tím je aproximace přesnější, ale výpočet pomalejší. Velikost dávky je také omezena velikostí paměti procesoru či grafické karty.

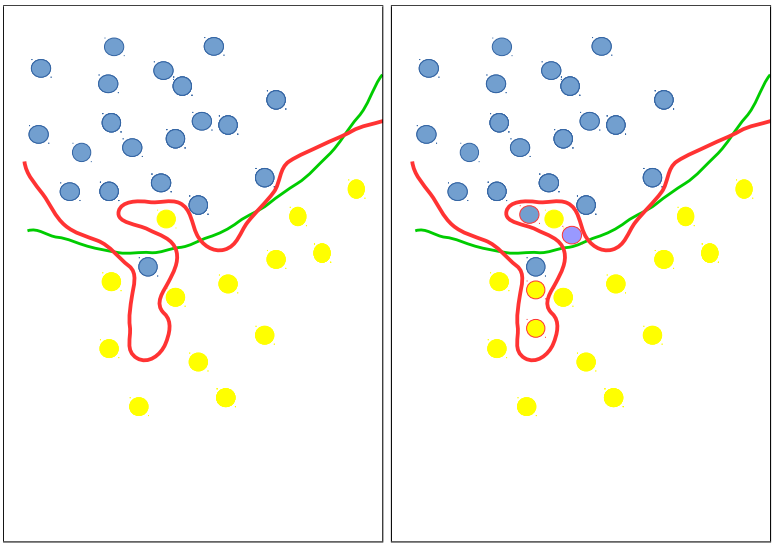
Rychlost a přesnost výpočtu také ovlivňuje velikost obrázku. Při návrhu sítě je určena velikost vstupního vektoru sítě. Je-li obrázek ve stupních šedi, pak každý pixel odpovídá jednomu vstupu. Pokud chceme, aby síť zpracovávala barevné obrázky, pak každý pixel odpovídá třem vstupům sítě[11]. Například barevný obrázek o rozměrech 227×227 je možné zpracovat sítí o 154587[12] vstupech. Před klasifikací je třeba obrázky upravit, aby odpovídali zadaným rozměrům. Čím větší rozměr obrázků zvolíme, tím více detailů zachováme. Zvyšujeme však dobu výpočtu a požadavky na paměť V současnosti se v Caffe [20] nejčastěji používají rozměry 227×227. V praxi se používají nejčastěji čtvercové obrázky.

Přeučení

Stejně jako u většiny algoritmů, které využívají strojové učení hrozí i u neuronových sítí riziko tzv. přetrénování. Cílem učení totiž není, aby síť poskytovala správné výsledky nejen na vstupy z datové sady, ale i na vstupy, z kterých se neučila. Cílem je, aby síť abstraho-

[7] backpropagation
[8] Pravděpodobně se nejedná o minimum
[9] batch
[10] Například ve vzorových implementacích sítí ve frameworku Caffe [20] se používá dávka 256 obrázků.
[11] Uvažuji obrázky se třemi kanály
[12] $227 \times 227 \times 3$

vala informace obsažená v trénovacích datech a bylo poté možné získat správné výstupy na libovolná vstupní data. To že síť vrací stále lepší výsledky na trénovacích datech totiž neznamená, že síť bude schopna správně zařadit nová data. Tento jev se nazývá přetrénování[13] a je ilustrován na obrázku 3.4. Přeučení se lze vyhnout oddělením testovacích a trénovacích dat a průběžným testováním. V okamžiku, kdy stále klesá objektivní funkce, ale výsledky na testovacích datech se nezlepšují či se dokonce zhoršují, je třeba zastavit trénování, protože právě dochází k přeučení. Jsou-li potřeba přesnější výsledky, pak je možné zkusit proces trénování opakovat s jinými počátečními vahami nebo změnit strukturu sítě. Dalším řešením je rozšířit datovou sadu, což je v některých případech velmi obtížné, protože to často vyžaduje ruční klasifikace mnoha obrázků.



Obrázek 3.4: Schéma přetrénování. Vlevo vidíme 2 rozdělující linie zástupci dvou tříd. Červená má vyšší úspěšnost na trénovacích datech. Není však dostatečně abstraktní, takže nově přidané vzorky (vpravo) klasifikuje lépe zelená linie.

Použití předem natrénované sítě

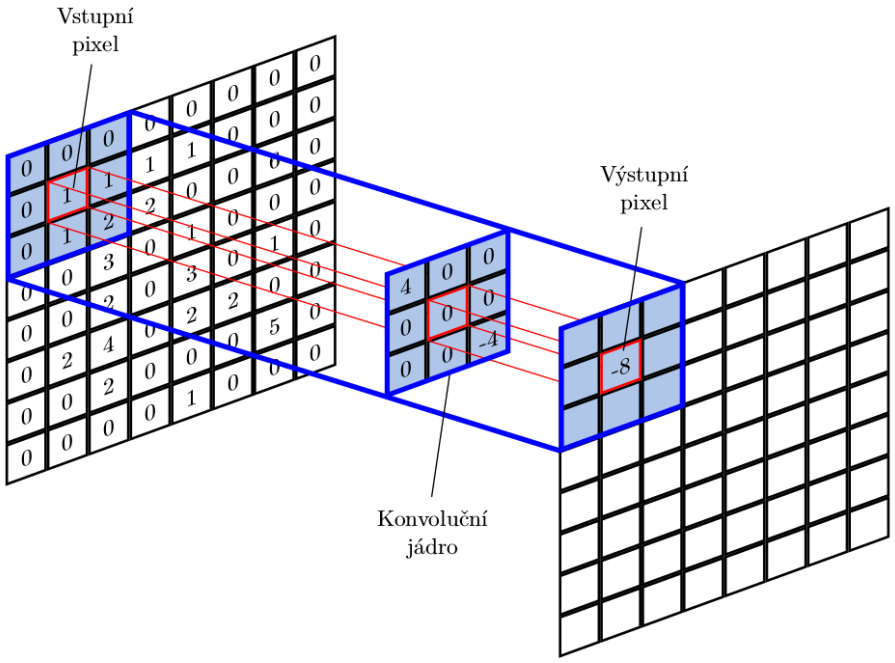
Při menších datových sadách lze použít neuronovou síť naučenou na řešení jiného ale podobného problému. Síť je pak upravena a dotrénována na datové sadě pro řešený problém[14]. Učení sítě potom vyžaduje méně iterací, čímž se proces zrychlí a sníží se riziko přetrénování sítě [8]. Aby bylo možné použít tento postup, je třeba aby síť měla stejnou strukturu. Pokud máme například naučenou síť, která klasifikuje obrázky do 1000 kategorií a chceme ji doučit, aby klasifikovala houby do 100 kategorií, musíme upravit strukturu sítě, aby poslední vrstva měla 100 neuronů. Potom se hodnoty všech synapsí nastaví podle už naučené sítě, jen hodnoty poslední vrstvy se nastaví náhodně.

3.3 Konvoluční sítě

Konvoluční neuronové sítě[23] (CNN) jsou jedním typem neuronových sítí, které se používají pro zpracování signálů například obrázků[46]. Konvoluční vrstva se skládá z naučených

[13] overfitting
[14] Anglicky se tento pustup nazývá finetuning

filtrů, které jsou aplikovány vstup jako konvoluční jádro. Schéma konvoluce je ilustrováno na obrázku 3.5.



Obrázek 3.5: Princip konvolučního filtru. Každý pixel na výstupu se vypočte jako vážený průměr svého okolí na vstupu. [14]

Tím, že se síť používá na zpracování obrázků, je možné využít některé vlastnosti obrázků. V popisu neuronových sítí jsme viděli, že síť se skládá z jednotlivých vrstev neuronů, kde každý neuron je spojen se všemi neurony předchozí vrstvy. Lze předpokládat, že pixely, které leží v obrázku blízko sebe, má smysl zpracovávat společně. Tomu odpovídá i architektura CNN, kde jsou neurony v jednotlivých vrstvách uspořádány ve třech dimenzích. Neurony jsou propojeny pouze s malou oblastí předchozí vrstvy . Vstupy a výstupy těchto vrstev jsou třírozměrné matice, kdy vstupní vrstva má rozměry, které odpovídají rozměrům obrázku a počtu kanálů. Výstupní vrstva má potom rozměry [1 x 1 x <počet kategorií>].

Tím, že jsou plně propojené vrstvy nahrazené konvolučními vrstvami, které používají pouze malá konvoluční jádra, mají CNN méně parametrů než klasické neuronové sítě [26].

CNN se začaly masově používat po úspěchu Alexandra Krizhevského v soutěži ILSVRC[15] v roce 2012.

3.3.1 Používané vrstvy

Nyní stručně popíšu vybrané vrstvy, které se v konvolučních sítích používají.

- **Konvoluční vrstva** Základní stavební blok neuronových sítí. Neurony ve stejné hloubce konvoluční vrstvy sdílí váhy a práh, proto je možné její výstup počítat jako konvoluci s jádrem skládajícím se z těchto vah. Odtud plyne název konvoluční neuronové sítě.

[15] ImageNet Large Scale Visual Recognition Challenge

- **Pooling vrstva** Mezi konvoluční sítě se vkládají pooling vrstvy, které snižují rozměry následující vrstvy. Provádí podvzorkování tak, že několik hodnot agreguje do jedné nejčastěji pomocí výběru maxima.

- **Plně propojená vrstva** Tato vrstva odpovídá vrstvám používaných v klasických neuronových sítích. Každý neuron této sítě je propojen s každým neuronem sítě předchozí. Obvykle se nachází na konci sítě.

- **Softmax** Používá se jako výstupní vrstva. Převede libovolný K-rozměrný vektor **z** na vektor stejných rozměrů s hodnotami v intervalu (0,1) se součtem 1. Její výstup tedy lze interpretovat jako pravděpodobnost [9].

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}; j \in 1..K \tag{3.3}$$

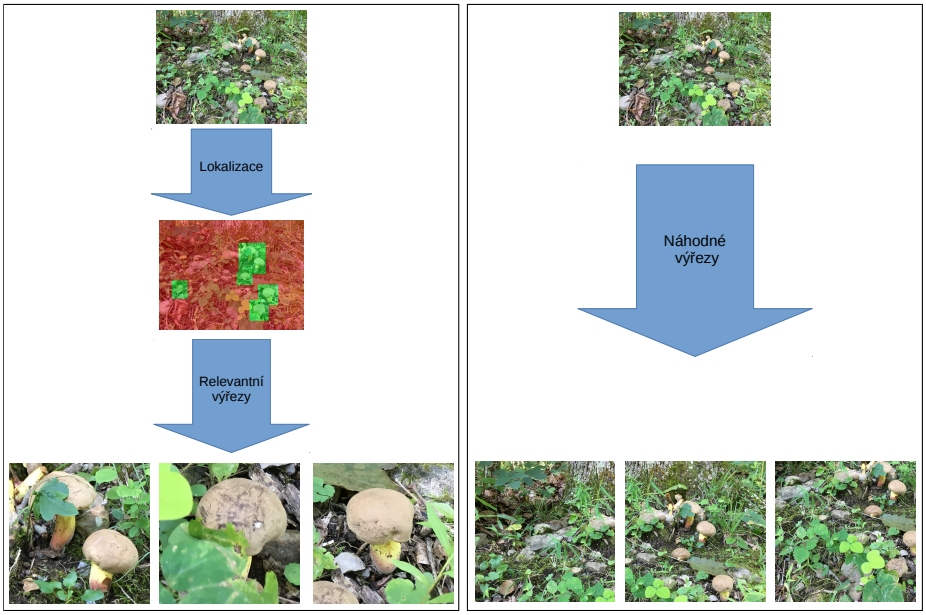
- **Normalizační vrstva** Jak název napovídá, vrstva normalizuje vstupní hodnoty. Vrstva potlačuje extrémní hodnoty.

- **Dropout** Jeden ze způsobů, jak se předchází přetrénování sítě je použití vrstvy dropout. Vrstva pouze zkopíruje data ze vstupu na výstup, s tím že určité náhodné procento signálů vynuluje. Její použití vede k robustnějšímu učení, protože síť se nemůže spoléhat na výstupy konkrétních neuronů.

Kapitola 4

Metody lokalizace hub

Jedním z úkolů mé diplomové práce byla detekce hub na fotografii v přirozeném prostředí. Původně jsem chtěl trénovat neuronovou síť na už segmentovaných obrázcích s vyznačenými jednotlivými částmi houby. Ukázalo se však, že problém lokalizace houby není triviální a použití tradičních metod nepřineslo očekávané výsledky. Stručně popíši s jakými algoritmy jsem experimentoval a jaká byla jejich úspěšnost. Nejvíce se budu věnovat lokalizaci pomocí neuronové sítě, která se ukázala jako nejúspěšnější.



Obrázek 4.1: Použití lokalizace pro zefektivnění učení sítě. Pokud známe polohu houby je možné trénovat síť nad relevantními daty. Navíc je možné vyhnout se deformaci obrázku v důsledku změny měřítka. Fotografie z [6]

Lokalizace je z hlediska mé práce důležitá především proto, že umožňuje lépe připravit datovou sadu. Má datová sada totiž obsahuje velké množství obrázků s rozměry v řádu tisíců pixelů, které je nutné před vstupem do sítě zmenšit na požadovanou velikost. To se provádí změnou měřítka, vyřezáváním nebo jejich kombinací, což vede u fotografií ke ztrátě informací a k deformacím. Bez předchozího určení lokalizace objektu dochází k tomu, že náhodně vyříznutá oblast určená k trénování sítě buď houbu vůbec neobsahuje, nebo obsahuje navíc velké množství šumu a houba často zabírá méně než 5% plochy fotografie.

Po zvládnutí lokalizace je možné obrázek vhodně zmenšit a vyříznou, aby každý obrázek obsahoval celou houbu a minimum pozadí, jak je ukázáno na obrázku 4.1.

V této kapitole se často odkazuji na kapitolu 5. Je to způsobeno tím, že pro lokalizaci hub jsem často používal výsledky popsané v kapitole 5 a naopak.

4.1 Metody, které vyžadují ruční anotaci

Pomocí knihovny OpenCV jsem vytvořil skript, který umožňuje velmi rychlou ruční anotaci, kdy vytvoření trimapy pro jeden obrázek zabralo pouze několik sekund. Vzhledem rozsáhlosti použité datové sady, se však stejně ukázalo, že není možné všechny obrázky ručně anotovat. Výsledky metod, které jsem takto anotoval byly však natolik dobré, že jsem je poté používal jako referenční obrázky pro měření přesnosti jiných lokalizačních metod.

4.1.1 Algoritmy Alpha-matting a slévání

Algoritmus Alpha-matting [18] či metoda slévání z knihovny OpenCV [39] (watershed), pracují na podobném principu. Pro svůj začátek vyžadují označení přibližné polohy objektu pomocí například tzv. *trimapy* a poté oblast s hledaným objektem segmentují. Oba tyto algoritmy poskytovali velmi dobré výsledky, ale nakonec jsem je nepoužil kvůli nutnosti vytvářet trimapy, což trvá pro každý obrázek několik sekund. Příklad výsledku je uveden na obrázku 4.2.

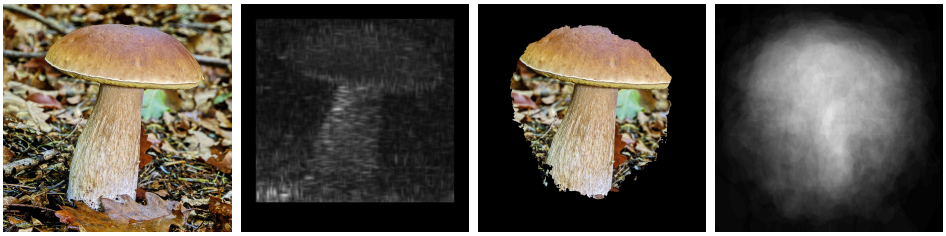
Obrázek 4.2: Obrázek houby, ručně vyvořená trimapa přibližně popisující její polohu a výstup algoritmu alpha-matting určující přesnou polohu houby. Fotografie z [6]

4.2 Metody bez nutnosti ruční anotace

Dále jsem experimentoval s hledáním zaostřených míst pomocí Fourierovi transformace. Část fotografií hub totiž byla zaostřena na hledanou houbu, zatímco pozadí bylo rozostřené. Hledal jsem oblasti na obrazu, kde je největší energie na vysokých frekvencí. Příklad fotografie, kde byla tato metoda úspěšná je na obrázku 4.3.

Dále je výskyt houby na fotografii nejčastěji uprostřed obrazu, vyzkoušel jsem proto použití metody řezu grafem, kdy jsem okraj obrazu označil jako pravděpodobné pozadí a střed jako pravděpodobný objekt. Tuto metodu jsem ještě zdokonalil výpočtem průměrného umístění houby.

Testy nakonec ukázali že žádná z těchto metod není použitelná.



Obrázek 4.3: Neúspěšné metody lokalizace bez nutnosti ruční anotace. Vlevo je fotografie houby, dále výstup vysokofrekvenčního filtru implementovaného pomocí Fourierovi transformace, výstup metody řezu grafem a průměrné umístění houby na fotografii, fotografie Michael Wood

4.2.1 Lokalizace na základě barvy

Lokalizace na základě barev má mnohá úskalí a je málokdy dostatečně přesná, aby bylo možné na ni spoléhat. Přesto jsem ji využil a v kombinaci s výsledky lokalizace pomocí neuronových sítí jsem získával vstupy určené pro učení neuronové sítě. Metoda lokalizace na základě barvy je použitelná pouze v případě, kdy je k dispozici dostatečné množství fotografií ze stejné třídy. Z obrázků jsem vygeneroval histogram výskytu barev a porovnával jej s histogramem vzniklým z náhodných fotografií různých druhů. Tak jsem změřil, jaké barvy na fotografiích daného druhu hub převažují. Lze odhadovat, že to budou barvy hub, které jsou foceny. Oblasti, kde tyto barvy převažují se potom označí jako oblast obsahujíc houbu. Výsledky jsou uvedeny na obrázku 4.4. Pokud ovšem nějaká houba roste vždy ve stejném prostředí[1], pak tato metoda nemůže polohu nijak určit.

Obrázek 4.4: Houba a její lokalizace na základě barvy, fotografie Kryšot [6]

4.2.2 Kaskáda slabých klasifikátorů

Často používaným nástrojem pro detekci objektů je kaskáda slabých klasifikátorů [41]. Použil jsem její implementaci ve knihovně OpenCV [39]. Klasifikátor však nebylo možné natrénovat. Pokusil jsem se proto vytvořit jednodušší klasifikátor, který by detekoval pouze houby jednoho druhu, konkrétně Hřibu Smrkového. Ani tento jednodušší klasifikátor však nebylo možné natrénovat při libovolném nastavení parametrů. Ukázalo se, že tento algoritmus nebo minimálně tato implementace není pro můj problém vhodná.

[1] Metoda například zcela selhala u hub, které vyrůstají z kůry stromů.

4.3 Lokalizace pomocí neuronové sítě

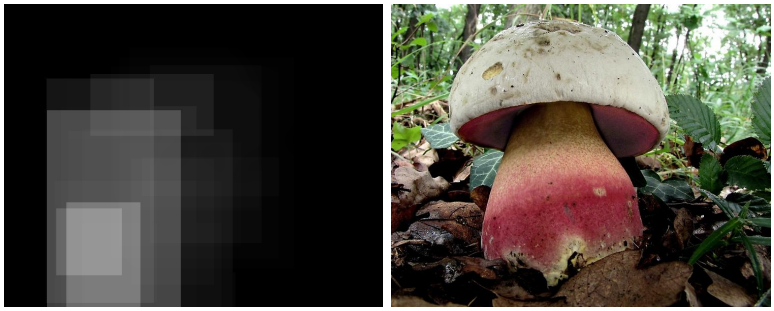
Existuje mnoho způsobů, jak využít neuronovou síť k lokalizaci hledaného objektu. Pokud bych měl k dispozici velké množství anotovaných obrázků, jaké obsahuje například datová sada Pascal [15]. Bylo by možné naučit neuronovou síť, aby přímo vracela polohu hledaného objektu. Já jsem však takovou datovou sadu k dispozici neměl a použití neuronové sítě naučené na lokalizaci obecných objektů nepřicházelo v úvahu kvůli zcela nedostatečným výsledkům. Jako nejvhodnější se proto ukázalo použití sítí naučených na klasifikaci. Provedl jsem to několika způsoby, které nyní dále rozvedu.

4.3.1 Lokalizace pomocí pohyblivého okna

K lokalizaci houby jsem použil klasifikaci pomocí pohyblivého okna. Z obrázku jsem postupně vybíral výřezy a u každého z nich rozhodoval, zda se na něm nachází hledaný objekt. Využil jsem k tomu klasifikaci celého obrázku.

Síť trénovaná na celých obrázcích

Při zpracování lokalizace hub jsem nejprve použil metodu posuvného okna[2] [25]. Síť se nejprve natrénuje síť na označených fotografiích. Potom, co síť umí klasifikovat podobné obrázky jsou síti předkládány výřezy fotografií a sleduje se odezva sítě . Lze předpokládat, že na výřezech, kdy má síť podobné výsledky jako při klasifikace, se nachází hledaný objekt [29]. Podobnost jsem měřil tak, že jsem vypočítal skalární součin vektorů pravděpodobností. Příklad výstupu je uveden na obrázku 4.5.



Obrázek 4.5: Houba a její lokalizace pomocí neuronové sítě, fotografie Kryšot [2]

Obrázek jsem nejprve klasifikoval neuronovou sítí. Poté jsem z obrázku vyřezával náhodné obdélníky a ty neuronovou sítí klasifikoval. Porovnával jsem podobnost získaných klasifikací a podle toho označoval obdélník jako objekt obsahující či neobsahující. Takto jsem provedl 100 iterací a získal přibližnou polohu houby.

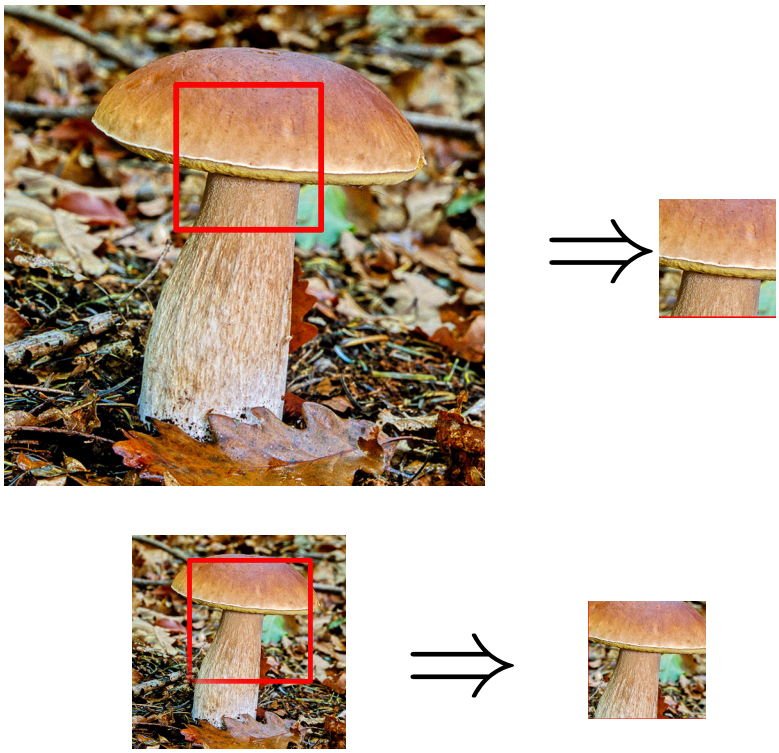
Síť trénovaná na výřezech z obrázků

Výše popsaný postup není optimální, protože síť trénovaná na klasifikaci celých obrázků není nejvhodnější na klasifikaci výřezů. Na trénování Sítě ImageNet, kterou jsem v upravené podobě používal ke klasifikaci se používali obrázky o rozměrech 256x256, ze kterých se náhodně vyřezávali vzorky o rozměrech 227x227. Pro účely klasifikace je však vhodnější síť

[2] anglicky sliding window

naučená na klasifikaci menších výřezů, protože při použití posuvného okna jsou jí na vstup přiváděny také pouze malé výřezy fotografií.

Velikost výřezů je pevně daná faktem, že naučené sítě, které byli k dispozici měli dané vstupní rozměry, co jsem mohl měnit byla velikost vstupních obrázků. Čím větší jsou vstupní obrázky, tím menší jeho část výřez obsáhne. Ztratí se tak informace o kontextu, ale zase se získá větší množství detailů. Záležitost je ilustrována na obrázku 4.6. Po několika experimentech byla u sítě AlexNet zvolena velikost 1500x1500 pixelů a u sítě GoogLeNet velikost 750x750 obě s velikostí výřezů 227x227.

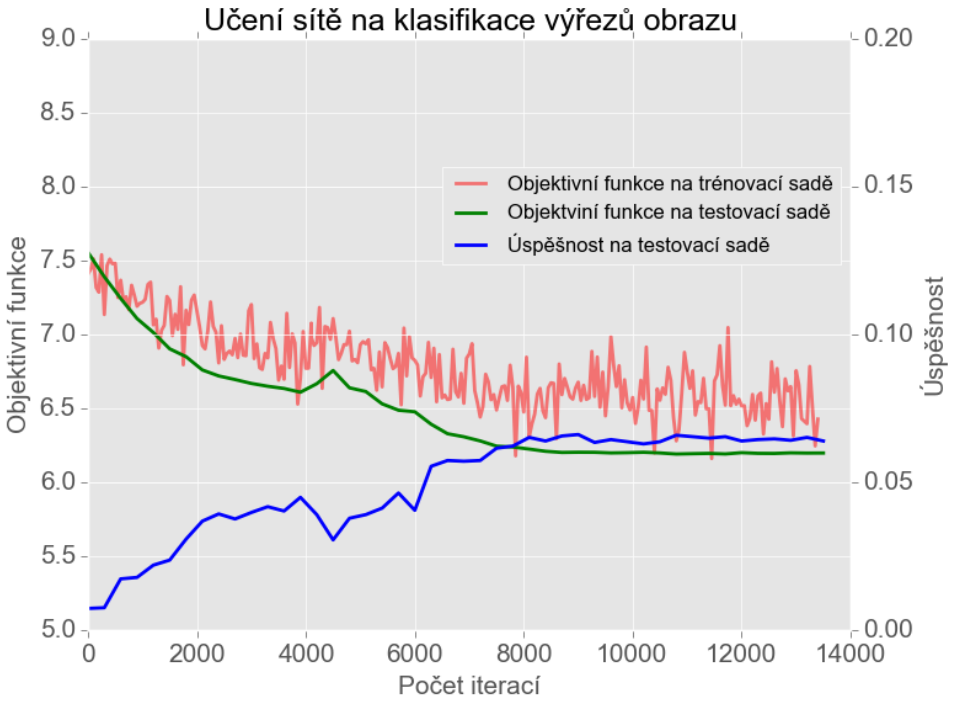


Obrázek 4.6: Vytváření výřezů pro učení neuronové sítě. U každého obrázku je nejprve změněno měřítko a poté je vyříznuto okno o pevné velikosti, které je vloženo na vstup sítě. Při vyřezávání z obrázku s velkým měřítkem zabírá výřez větší část houby, naopak při vyřezávání z obrázku s malým měřítkem, obsahuje více detailů. Fotografie Michael Wood

Byly natrénovány dvě sítě na klasifikaci výřezů obrázků. Úspěšnost sítě GoogLeNet se pohybovala okolo 10%, úspěšnost sítě AlexNet okolo 6%. Průběh učení sítě AlexNet je znázorněn na obrázku 4.7. Výstupem sítě pro každý výřez je opět klasifikace daného výřezu. V příští kapitole bude popsáno, jak je možné tuto síť využít pro lokalizaci houby.

4.3.2 Plně konvoluční sítě

Na začátku jsem pro lokalizaci použil síť naučenou na rozpoznávání na úrovni druhů. Použil jsem plně konvoluční sítě(FCN) podle článku [28]. CNN se převede na FCN tak, že se plně propojené vrstvy zamění za konvoluční vrstvy, jak je znázorněno na obrázku 4.8. Dále se zvětší velikost vstupního obrázku. Síť potom simuluje klasifikaci pohyblivého okna, které se posouvá po vstupním obrázku a klasifikuje označený výřez. Výstupem sítě potom není vektor pravděpodobností nýbrž třírozměrná matice hodnot, které přiřazuje pravděpodobnosti



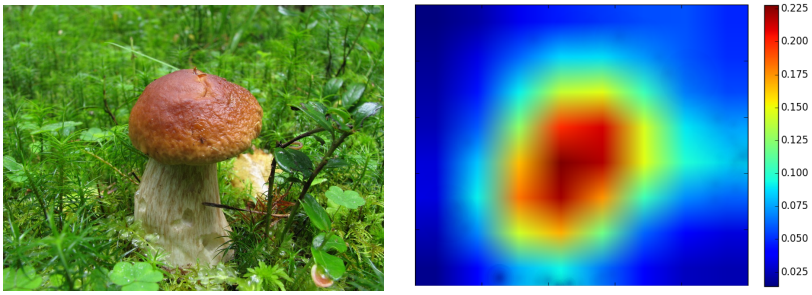
Obrázek 4.7: Průběh učení sítě trénované na výřezy obrázků.

(a) Schéma klasické konvoluční neuronové sítě, jejíž výstupem je vektor pravděpodobností

(b) Schéma plně konvoluční neuronové sítě, jejíž výstupem je třírozměrné pole pravděpodobností

Obrázek 4.8: Srovnání konvoluční a odpovídající plně konvoluční neuronové sítě.

jednotlivých kategorií každému výřezu. Jedná se o mapu klasifikací navzájem se překrývajících segmentů. Výpočet pomocí plně propojené neuronové sítě je násobně efektivnější než opakovaná klasifikace jednotlivých segmentů pomocí původní sítě [28].

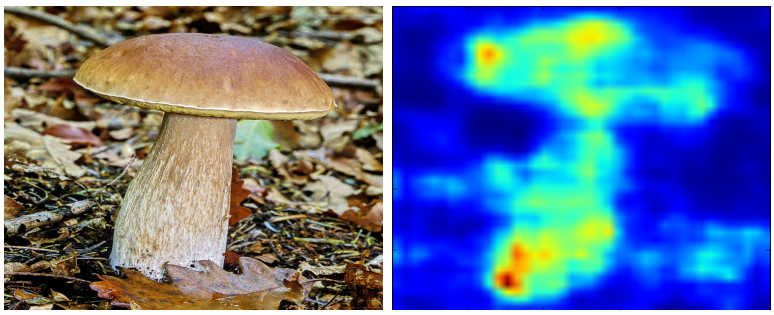


(a) Hřib Smrkový

(b) Měřítko udává výsledek skalárního součinu

Obrázek 4.9: Houba a její lokalizace pomocí plně konvoluční neuronové sítě bez předchozí znalosti kategorie. Fotografie Mvkarpov Toadstool.ru

Polohu hledaného objektu potom získám porovnáním klasifikace každého segmentu s klasifikací původního obrázku. Provedl jsem to tak, že jsem provedl maticové násobení získané matice s vektorem pravděpodobností získaných při klasifikaci celé fotografie. Příklad je uveden na obrázku 4.9.



Obrázek 4.10: Houba a její lokalizace pomocí plně konvoluční sítě se znalostí kategorie houby, fotografie [6]

.

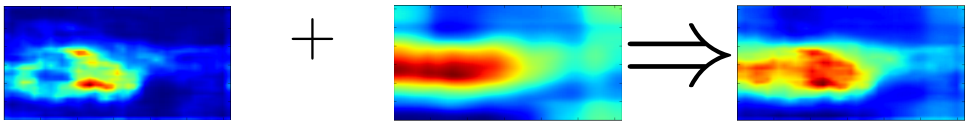
Jedná-li se navíc o lokalizaci na obrázku z datové sady, je možné použít znalost do které kategorie daná fotografie patří. Pak se u každého výřezů zkoumá shoda přímo s kategorií zpracovávané fotografie, čímž se dosahuje větší přesnosti. Příklad je na obrázku 4.10.

Postup lokalizace probíhá tak, že se obrázek přivede na vstup obou FCN. Získají se takto dvě teplotní mapy popisující polohu houby. V nic obsaženou informaci je možné zkombinovat mnoha způsoby. Zvolil jsem aritmetický a geometrický průměr. Celý postup je ilustrován na obrázku 4.13

4.4 Vyhodnocení úspěšnosti lokalizace

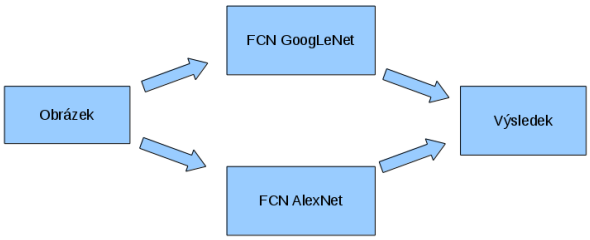
Pro každý obrázek jsem tedy získal celkem 4 teplotní mapy, určující polohu houby, jak je vidět na obrázku 4.12. Pro účely vyhodnocení úspěšnosti jsem obrázky prahoval. Experimentoval jsem s různými hodnotami prahu získanými různými kombinacemi mediánu,

Obrázek 4.11: Obrázek houby, fotografie [6]

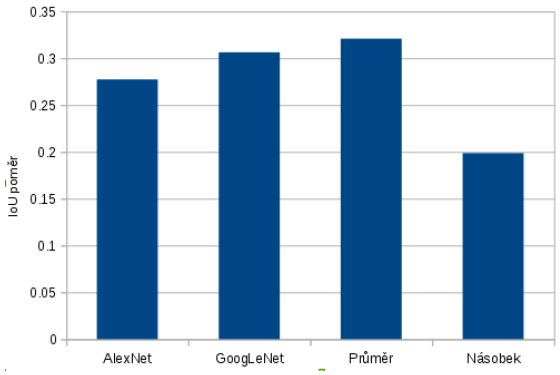


Obrázek 4.12: Vlevo je lokalizace houby sítí AlexNet, uprostřed lokalizace sítí GoogLeNet, napravo kombinace těchto výstupů prostým aritmetickým průměrem.

aritmetického průměru a směrodatné odchylky, ale nejlepších výsledků se dosáhlo pomocí prostého aritmetického průměru. Výsledek prahování se pak porovnával s referenční polohou houby. získanou z trimapy algoritmem alpha-matting, jak popisuje kapitola 4.1.1. Úspěšnost jsem porovnával pomocí metriky *intersection over union* [36], která se používá pro hodnocení výsledků lokalizace. Úspěšnost se vyhodnotí jako poměr průniku ku spojení odhadované a skutečné polohy objektu.



Obrázek 4.13: Schéma výsledného návrhu aplikace pro lokalizaci hub. Každý obrázek vyhodnotí každá síť zvlášť a výsledek se potom zkombinuje.



Obrázek 4.14: Znázornění výsledků jednotlivých lokalizačních metod pomocí metriky *Intersection over union*.

Teplotní mapy znázorňující polohu houby na obrázku je možné kombinovat více způsoby, nejlepších výsledků bylo dosaženo pomocí prostého průměru teplotní mapy obou sítí, který

překonal výsledky obou sítí jednotlivě, jak je znázorněno na grafu 4.14. Je možné, že existuje lepší způsob, jak kombinovat teplotní mapy. Lokalizace není dostatečně přesná na to, aby ji bylo možné použít pro předzpracování obrázků před trénování neuronové sítě. Na rozdíl od klasifikace rostlin v Oxfordu [33] byla tedy síť trénována na celých obrázcích bez předchozí segmentace.

Kapitola 5

Metody klasifikace hub

Výpočty jsem prováděl na počítači s následujícími parametry.

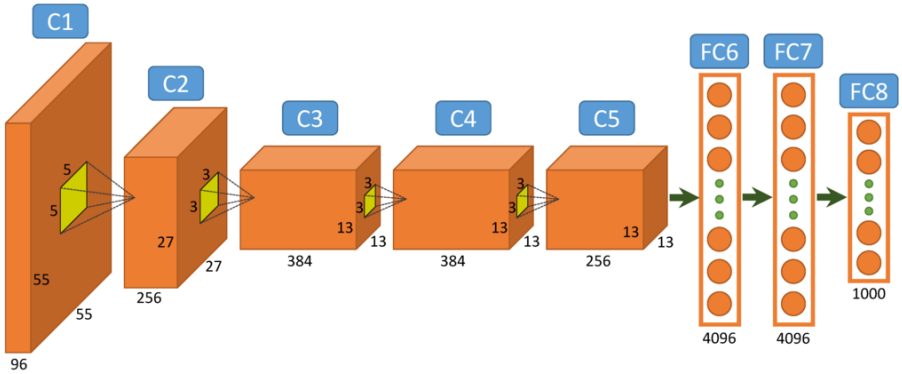
- CPU: AMD FX-8320E Eight-Core @ 3.2GHz
- GPU: GeForce GTX 1050 Ti, GeForce GT 740
- RAM: 16059MiB

Velikosti paměti jsem musel přizpůsobit velikost dávky, aby se všechna potřebná data vešla na kartu. Určil jsem proto velikost dávky 64 obrázků pro trénování zatímco všechny příklady sítí poskytované na stránkách projektu Caffe používají 256 obrázků.

Na připravených datech jsem natrénoval několik klasifikátorů a porovnal dosažené výsledky. Natrénoval jsem nejprve síť, která řadí houby do kategorií podle rodů a síť, která houby řadí podle druhů. Ke trénování jsem použil 482 618 obrázků.

5.1 Použité sítě

Jelikož jsem trénoval několik sítí ke klasifikaci obrazu, bylo přirozené použít již natrénované sítě a pouze je doučit k požadovanému úkolu. Použil jsem síť AlexNet jako představitele klasického návrhu konvoluční sítě a rozsáhlejší síť GoogLeNet vítěze soutěže ILSVRC v roce 2014.



Obrázek 5.1: Schéma CNN AlexNet. Poslední vrstva obsahuje v původní síti 1000 neuronů. Tento pčet byl změněn na 5097 pro klasifikaci na úrovni druhů respektive 299 pro klasifikaci na úrovni rodů [3].

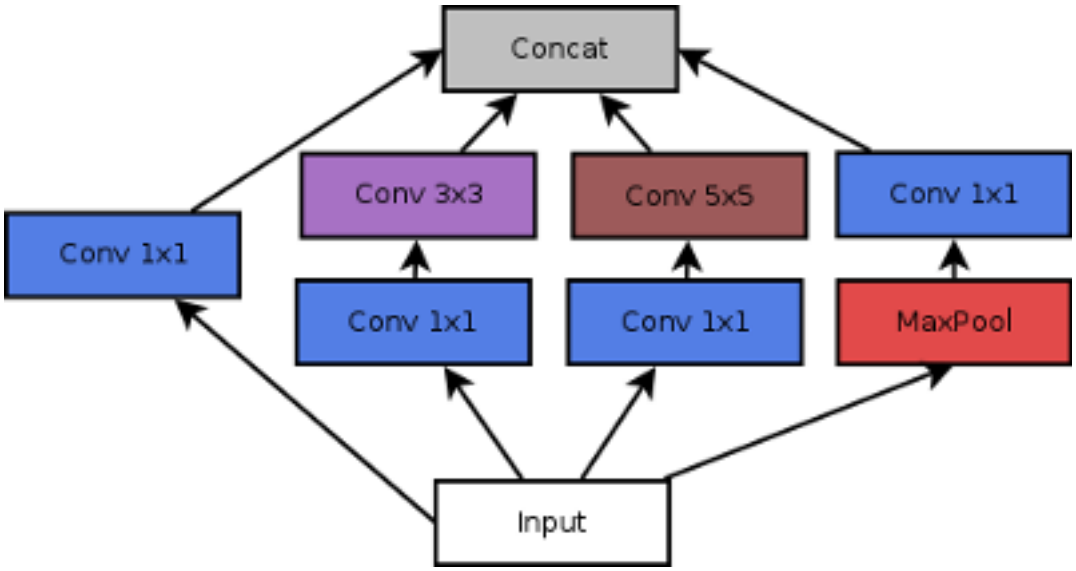
5.1.1 AlexNet

Před použitím jsou obrázky zmenšeny na velikost 256x256 při zachování tří kanálů. Poté je na vstup sítě přiváděn pokaždé jiný výřez o velikosti 227x227. Schéma sítě AlexNet je představeno na obrázku 5.1. Upravil jsem síť AlexNet, tak aby měla odpovídající počet výstupů a použil dvě její natrénované verze, které jsem dotrénoval na svých datech. První je [4] trénovaná na obrázcích z databáze ImageNet a druhá[33] je navíc dotrénovaná na databázi fotografií květin Oxfordské univerzity. Výsledky na obou sítí jsou po 40000 iteracích neodlišitelné. Je to pravděpodobně dáno tím, že druhá z výše jmenovaných sítí byla dotrénována pouze na 8189 obrázcích, což jsou pouze 2% mé datové sady. Dále jsem proto pracoval pouze s váhami získanými dotrénováním sítě z Oxfordské univerzity.

Síť dosáhla úspěšnosti 16%. Detailní vyhodnocení kvality klasifikátoru se nachází v kapitole 8.

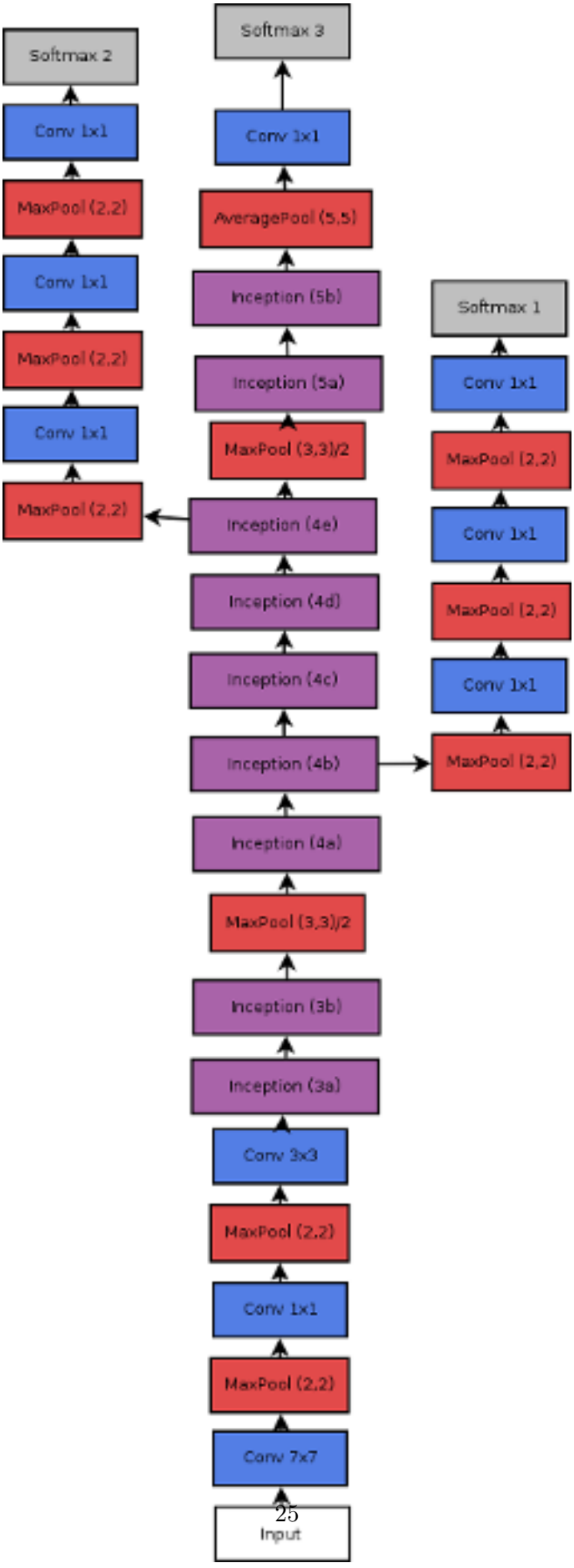
5.1.2 GoogLeNet

Další použitou sítě je GoogLeNet [38], vítěz soutěže ILSVRC 2014. Návrh sítě je nelineární, jak je vidět na obrázku 5.3. Základ sítě tvoří devět modulů nazývaných *inception*. Jedná se o nelineární propojení vrstev, kdy se vstupní data přivedou na několik konvolučních vrstev s různou velikostí konvolučních jader a jejich výsledky se potom konkatenují. Schéma je uvedeno na obrázku 5.2.

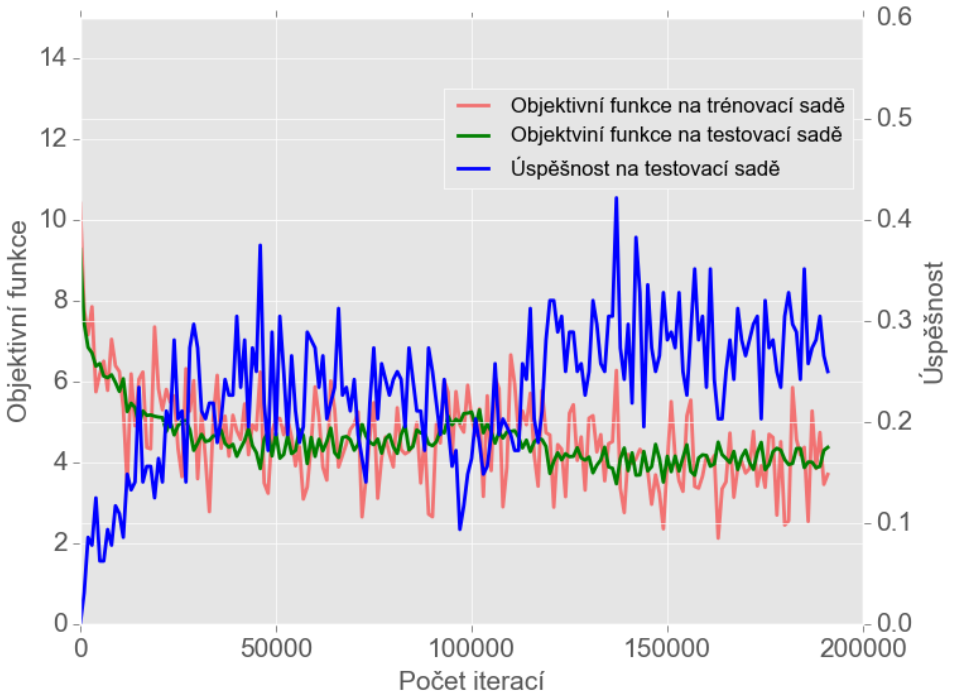


Obrázek 5.2: Schéma modulu *inception* CNN GoogLeNet. Zpracování probíhá nelineárně a výsledky jednotlivých konvolucí jsou poté konkatenovány [10].

Pro správné naučení sítě je nutné správně inicializovat váhy. I potom zabralo učení sítě GoogLeNet přibližně pětkrát více času než učení sítě AlexNet. Průběh učení je znázorněn na obrázku 5.4.



Obrázek 5.3: Schéma CNN GoogLeNet. Použítá vrstva inception je zobrazena na obrázku 5.2. Při učení se používají tři objektivní funkce vycházející z vrstev Softmax 1-3. Pro samotnou klasifikaci se používá pouze vrstva Softmax 3. [10]

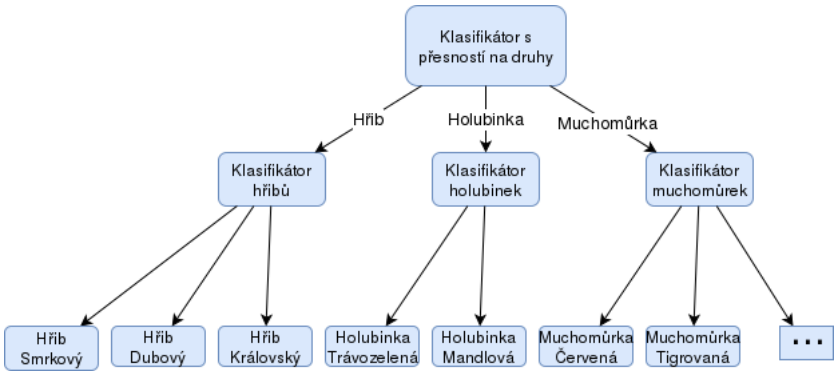


Obrázek 5.4: Průběh učení sítě GoogLeNet

Síť dosáhla úspěšnosti 25%. Detailní vyhodnocení kvality klasifikátoru se nachází v kapitole 8. Pro malou úspěšnost klasifikace na rody u sítě AlexNet a velkou časovou náročnost nebyla síť GoogLeNet trénování pro klasifikace na úrovni rodů. Je pravděpodobné, že delší trénování by vedlo ke zlepšení výsledků sítě.

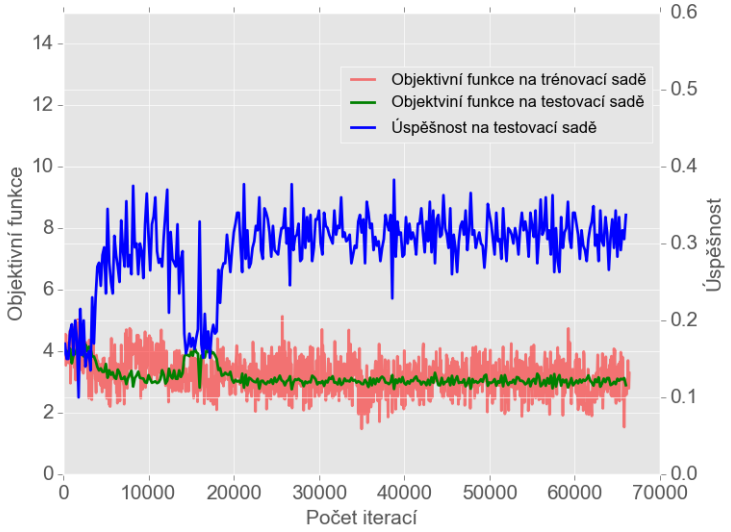
5.2 Síť klasifikující s přesností na rody

Při klasifikaci do hierarchicky uspořádaných kategorií připadá v úvahu vytvořit klasifikátor, který rozdělí vstupy do několika základních kategorií a dále klasifikátory, které budou naučené k rozlišování pouze v rámci své kategorie, jak je znázorněno na obrázku 5.5.



Obrázek 5.5: Schéma hierarchie Klasifikátorů

Síť AlexNet byla naučena na klasifikaci pouze na úrovni rodů. Její učení už nezlepšovalo úspěšnost, jak je vidět na obrázku 5.6, takže bylo ukončeno. Výsledky nepřekonali sítě naučené na klasifikaci na úrovni druhů, takže použití hierarchické klasifikace ztratilo smysl.

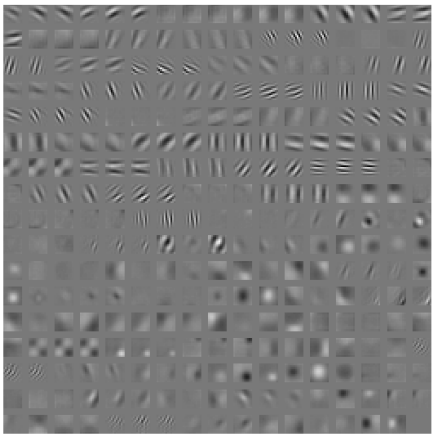


Obrázek 5.6: Záznam trénování sítě pro klasifikaci s přesností na rody

5.3 Vizualizace

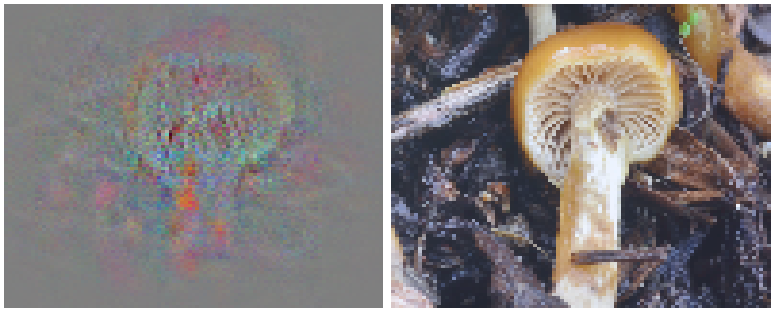
Podle článku [44] byly vizualizovány některé vlastnosti naučených sítí. Použity byly implementace [37] a [45].

První vizualizace 5.7 zobrazuje filtry v první konvoluční vrstvě s rozměry 11x11 pixelů.



Obrázek 5.7: Vizualizace filtrů první konvoluční vrstvy

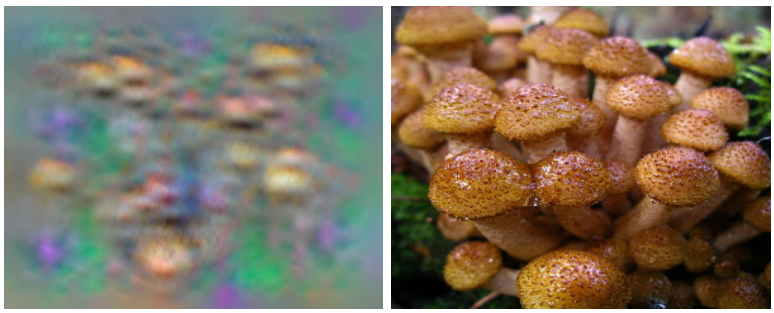
Další obrázek vizualizuje třetí vrstvu sítě AlexNet, dotrénovanou na klasifikaci hub jak vidíme na obrázku 5.8. Filtr nápadně připomíná vzhled některých hub.



Obrázek 5.8: Vizualizace filtru třetí konvoluční vrstvy a obrázek houby, kterou připomíná. Fotografie je volné dílo.

.

Dále byly podle výše zmíněného článku vygenerovány obrázky, které maximalizují předpověď sítě pro určenou kategorii. Jde určitým způsobem o vizualizaci toho, jak si síť představuje danou kategorie. Příklad výsledku je uveden na obrázku 5.9.



Obrázek 5.9: Obrázek maximalizující předpověď pro kategorii Václavka Obecná srovnaný se skutečnou fotografií této houby, Fotografie je volné dílo

.

Kapitola 6

Použitá data

Fotografie hub jsem stahoval z volně dostupných databází a stránek na internetu[1], získaná data jsem nahodile ručně kontroloval. Celkem jsem měl k dispozici přibližně 800 000 obrázků hub. Jednalo se téměř o 800 GB dat, takže veškerá manipulace s nimi byla náročná. To bylo také důvodem k tomu, že jsem datovou sadu ještě nerozšířil, ačkoliv na uvedených volně dostupných zdrojích byl k dispozici ještě alespoň dvojnásobek obrázků oproti použitému množství . Všechny informace o vybraných rodech či druzích, o kanonických jménech či synonymech najdete v odevzdaných souborech ve složce *Informace o datové sadě*.

6.1 Biologická taxonomie hub

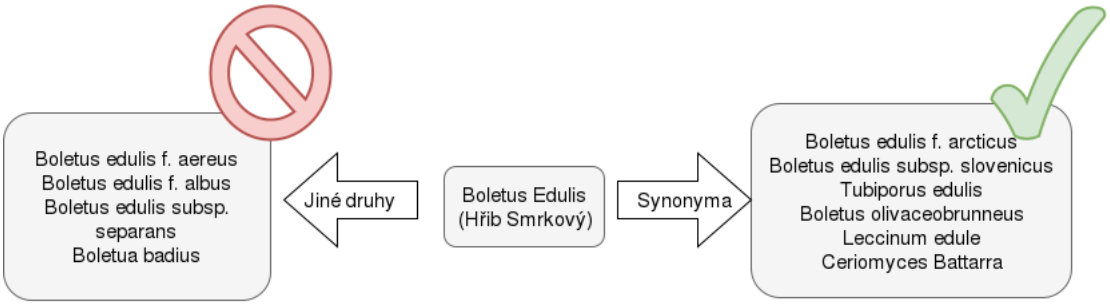
Tuto kapitolu jsem konzultoval s RNDr. Vladimírem Antonínem CSc, vedoucím botanického oddělení v Moravském zemském muzeu. S doktorem Antonínem jsem konzultoval i jiné informace týkající se mykologie, které budu uvádět dále.

6.1.1 Vyřešení synonym

Jak v českých tak i v latinských názvech není taxonomie jednotná, takže mají houby často mnoho názvů synonym. Zatímco českých synonym bývá výjimečně více než deset v latině jich bývají desítky. Je proto potřeba vybrat jedno tzv. kanonické jméno druhu a všechny fotografie označené jménem některého synonyma přeznačit. Protože jsem fotografie měl označené i zahraničními jmény, pokusil jsem se k setřídění použít data z wikipedie a upravit zařazení hub skriptem třídič. Bohužel jsem zjistil, že data na wikipedii obsahují chyby, protože docházelo ke zjevně nesprávným zařazením. Po konzultaci s mykologem jsem se rozhodl použít volně dostupná data se serveru Species Fungorum [5], poskytnutá Mezinárodní mykologickou asociací. Data z wikipedie jsem nakonec použil pouze na zařazení hub s jiným názvem než latinským.

Pro rozhodnutí, zda se jedná o synonymum nebo ne, je často potřeba znát název poddruhu nebo variace, ten však ve zdrojích, odkud jsem čerpal často nebyl uveden. Přestože jsem postupoval v souladu s terminologií používanou Mezinárodní mykologickou asociací, jsem při nahodilé kontrole datové sady objevil, že některé houby jsou řazeny do špatných kategorií. Podle subjektivního odhadu založeného na kontrole vybraných druhů, ale jejich počet nepřesahuje 1%.

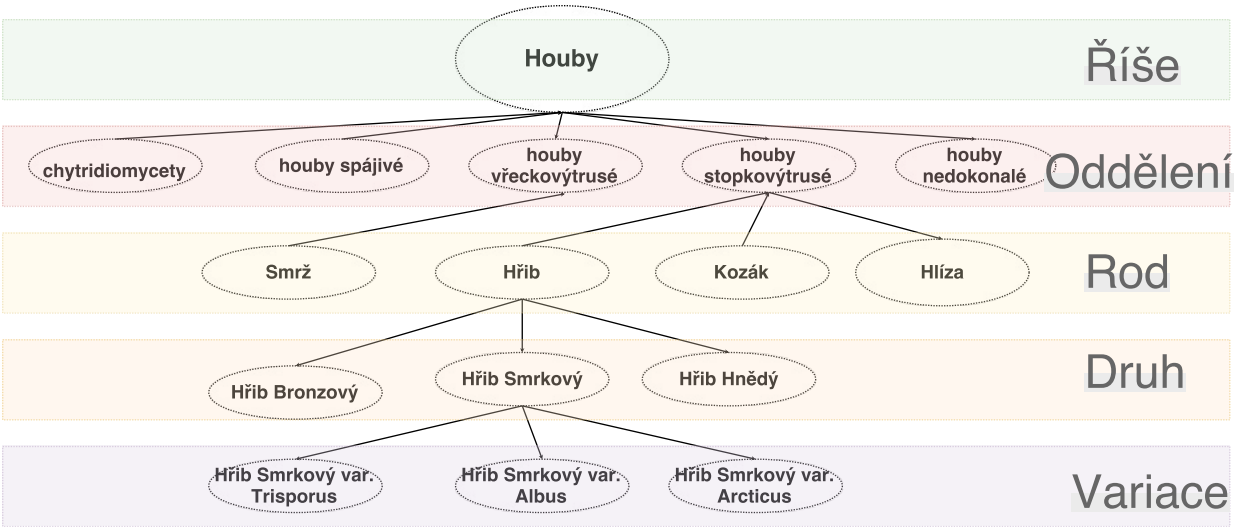
[1] Jejich seznam naleznete v odevzdaném CD.



Obrázek 6.1: Znázornění problému synonym. Rozhodnutí, zda se jedná o synonymum často závisí až na určení poddruhu či variace, které ale často nejsou uváděny.

6.1.2 Úroveň klasifikace

Biologie řadí organismy do stromově uspořádaná struktury, jak je znázorněno na obrázku 6.2. Říše hub se tak dělí na 6 základních oddělení, oddělení se dělí na rody, ty pak na druhy, poddruhy a odrůdy[2]. [47] Klasifikovat houby podle nižších stupňů než jsou rody je v praxi nepoužitelné. V tedy úvahu přichází tři možnosti klasifikace: podle rodů, druhů nebo odrůd. Moje datová sada neumožňuje klasifikovat podle odrůd, protože v drtivé většině mých zdrojů byly houby klasifikovány pouze do úrovně druhů. Mohu tedy klasifikovat podle rodů, kde bych tak měl k dispozici 915 kategorií, nebo podle druhů, kde mám 11023 kategorií.



Obrázek 6.2: Zjednodušená taxonomie hub

6.1.3 Výběr hub ke klasifikaci

Ne všechny druhy hub byly skutečně vhodné pro mou klasifikaci, protože z biologického hlediska se za houby považují například i různé plísně, které nejsou pro běžné houbaře použitelné, jak je možné vidět na obrázku 6.3. Rozhodl jsem se, že budu klasifikovat pouze druhy, které by mohly být využitelné pro případné houbaře, tedy houby, které mají viditelnou plodnici. Výběr byl také přizpůsoben tak, aby ke každému klasifikovanému druhu bylo k dispozici nejméně 100 různých fotografií. Druhy, kde jsem měl méně než uvedený

[2]Přehled je záměrně zjednodušený

počet jsem použil pouze pro klasifikaci na základě rodu. Nakonec jsem vybral 5097 druhů a 299 rodů, na kterých jsem síť učil a trénoval. Po této redukci zpracovávaných obrázků a po odfiltrování nevhodných, nepoužitelných či poškozených obrázků zbylo 536 271 obrázků, které jsou vhodné pro vyvíjenou aplikaci. Z toho jsem 10% použil k testování.

Obrázek 6.3: Vybraná houba s plodnicí a nevybraná plíseň, fotografie James Lindsey commanster.eu

6.1.4 Odstranění fotografií v mikroskopu

Protože jsem si data vytvořil sám obsahovali také nevhodné vzorky. Kromě Prázdných, špatně formátovaných či nekompletních obrázků se jednalo především o snímky z mikroskopu, či o různé kresby, které se fotografiím hub nijak nepodobají a nepřispívají tedy k učení neuronové sítě, jak je ilustrováno na obrázku 6.4. Vadná a špatně formátované obrázky byly odstraněny pomocí skriptu *cistic.py* Obrázků z mikroskopu bylo v datasetu přibližně 2-5%.

Postup probíhal iterativně. Nejprve jsem bylo pomocí skriptu *filtrMikroskop.py* ručně vyhledáno 350 obrázků mikroskopu a cca 3000 obrázků hub z lesa. Na tomto vzorku jsem natrénoval první klasifikátor, který měl při správně nastaveném prahu false positive menší než 50%. Tímto klasifikátorem jsem v datové sadě vyhledal 5000 obrázků z mikroskopu a 10000 fotografií ve skutečné velikost. Ručně jsem je zkontroloval a na těchto datech natrénoval už definitivní klasifikátor, kterým jsem data očistil.

Současný podíl vadných dat odhaduji na méně než 1%. Vycházím ale pouze z ruční kontroly náhodných fotografií.

Obrázek 6.4: Obrázky hub z přírody a mikroskopu. Pro lepší výsledky klasifikace bylo nutné je v datové sadě odlišit a obrázky z mikroskopu vymazat, fotografie Michael Woods

Kapitola 7

Implementace

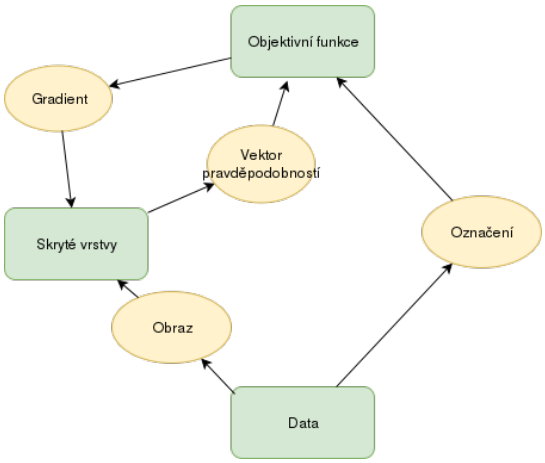
Tato kapitola popisuje nástroje, které byly při práci použity. Protože převážná část řešení používá konvoluční neuronové sítě, bude největší pozornost věnována právě jim.

7.1 OpenCV

Na začátku práce jsem nastudoval knihovnu OpenCV [39], která je velmi rozšířená a poskytuje implementaci základních algoritmů počítačového vidění. Použil jsem mimo jiné kaskádový klasifikátor LBP. Výsledky ale nebyly dostatečné, takže jsem tuto cestu musel opustit.

7.2 Caffe framework

Framewok caffe slouží k návrhu a trénování neuronových sítí. Výpočet probíhá podle algoritmů popsaných v kapitole 3. Na vstup je přiveden obrázek, je vypočtena odezva sítě a zpětně šířena chyba, jak ukazuje obrázek 7.1. Framework Caffe je psaný v C++/CUDA a ob-



Obrázek 7.1: Schéma výpočtu ve frameworku Caffe. Nejprve se v dopředném průchodu vypočítá klasifikace obrázku, ta se srovná se skutečnou klasifikací obrázku a zpětně se šíří chyba.

sahuje už zkompilované programy. Je však implementováno i rozhraní do jazyků MATLAB a python, které jsem používal.

Neuronové sítě se ukázaly jako ideální metoda. Učení neuronových sítí je však výpočetně náročná operace a ukázalo se, že obyčejný procesor není dostatečně rychlý. Proto bylo potřebné použít framework Caffe, který používá prostředí CUDA a k výpočtům používá grafickou kartu. Přepínání mezi výpočtem na CPU a na GPU se nastaví pomocí jediné proměnné. V mém případě se na GPU provedlo ve stejném čase 12krát více iterací než na CPU.

Implementace v prostředí Caffe se provádí textovým popisem sítě ve formátu Google protobuf [40], kdy se pouze popíše struktura a parametry jednotlivých vrstev a není nutné samotný výpočet implementovat. Příklad implementace poslední vrstvy ukazuje níže uvedený zdrojový kód.

```
layer {
    name: "fc8houby"
    type: "InnerProduct"
    bottom: "fc7"
    top: "fc8houby"
    param {
        lr_mult: 10
        decay_mult: 0
    }
    inner_product_param {
        num_output: 5097
        weight_filler {
            type: "xavier"
            std: 0.01
        }
        bias_filler {
            type: "constant"
            value: 0
        }
    }
}
```

Obrázek 7.2: Příklad popisu plně propojené vrstvy ve frameworku Caffe

7.2.1 Reprezentace dat

Data jako jsou zpracovávané obrázky, parametry modelu nebo vypočtená chyba se předávají pomocí struktury zvané Blob, která je čtyřrozměrnou maticí s rozměry (batch x výška x šířka x hloubka) uloženou v paměti.

Vstupní data je možné načítat přímo ze souboru nebo z paměti. Doporučuje se ale použití specializované databáze ve formátu LMDB či LevelDB, což zrychluje načítání dat a minimalizuje počet přístupů na disk. Ve své práci jsem použil formát LMDB [12].

7.2.2 Závislosti

Framework Caffe je rozsáhlý, proto vyžaduje několik jiných balíčků.

- **CUDA, OpenCV**
- **BLAS** Knihovna pro efektivní výpočet maticových operací. Použil jsem implementaci Atlas [43]. Atlas umožňuje automaticky optimalizovat svůj výkon na daném hardwaru tak, že změří náročnost jednotlivých instrukcí, a potom používá ty instrukce, které jsou nejméně náročné.
- **Boost**
- **Glog,gflags** Zajišťuje logování souborů a umožňuje pracovat v příkazovém řádku.
- **LevelDB, LMDB** Databáze obrázků, která zrychluje jejich načítání a optimalizuje množství přístupů na disk.
- **Protobuf** Formát na definování datových struktur. Soubory v tomto formátu mohou být například zkompilovány do hlavičkových souborů v C++.

7.2.3 CUDA

Pro efektivní trénování je nutné využít GPU. Při začátku práce jsem sítě trénoval na procesoru *Intel(R) Pentium(R) CPU 2117U @ 1.80GHz*. Po přechodu na grafickou kartu *GeForce GTX 1050 Ti* se proces učení zvýšil sedmínásobně. Framewok Caffe umožňuje využít prostředí CUDA, což je softwarová architektura umožňující spouštět procesy na GPU firmy nVIDIA [24].

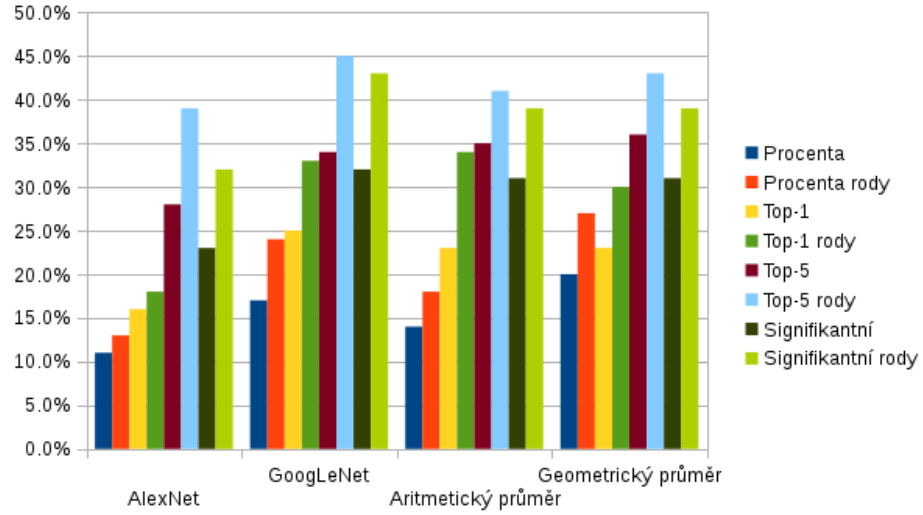
Kapitola 8

Vyhodnocení výsledků

V kapitole 5 byl popsán učení sítí, na tomto místě představím jakých výsledků tyto sítě dosáhly na testovacích datech. Ke zhodnocení použiji několik metrik, které dále vysvětlím. Síť klasifikující s přesností na druhy je vždy zároveň převedena na síť klasifikují s přesností na rody tak, že jsou pravděpodobnosti druhů patřících do stejného rodu sečteny. K hodnocení jsem použil následující metriky [34]:

- **Úspěšnost**, neboli top-1 úspěšnost je nejčastěji používaným způsobem měření kvality klasifikátoru. Získá se jako procento případů, kdy byla fotografie správně klasifikována.
- **Top-5 úspěšnost** [17] se získá jako procento případů, kdy byla správná kategorie označena jako jedna z pěti nejpravděpodobnějších. Její použití je vhodné, pokud bude výsledná aplikace uživateli zobrazovat 5 nejpravděpodobnějších tříd.
- **Procenta přiřazená správné kategorii** [11]. Neuronová síť přiřadí každé kategorii pravděpodobnost, je možné měřit, kolik procent bylo průměrně přiřazeno správné kategorii. Pro uživatele je totiž důležité vědět, za jak jistou může klasifikaci považovat.
- **Statistická významnost**. Možnost, že objekt je určité kategorie označím za statisticky významnou, pokud mu klasifikátor odhadl více než 5%. Tato metoda měření kvality klasifikátoru bude nejvhodnější, pokud bude výsledná aplikace nastavena, aby zobrazovala všechny kategorie, které přesáhly mez 5%.

8.1 Vyhodnocení na celé datové sadě



Obrázek 8.1: Vyhodnocení úspěšnosti natrénovaných sítí na testovacích datech. Jsou uvedeny výsledky dvou naučených sítí a dvě možnosti, jak jejich výsledky kombinovat.

Na náhodně vybraných obrázcích, které nebyly použity ke trénování sítě jsem síť otestoval. K testování jsem si vyhradil 13 653 náhodně vybraných fotografií. Výsledky testu jsou znázorněny na grafu 8.1. Ukazuje se, že síť GoogLeNet je výrazně úspěšnější a žádný způsob, jak získané výsledky obou sítí kombinovat nepřekonává výsledky samotné sítě GoogLeNet.

Zkoumaná problematika je ale složitější a obvyklý způsob měření úspěšnosti není pro vyhodnocení mé aplikace optimální z následujících důvodů.

- Drtivá většina houbařů sbírá nejvýše 100 různých druhů hub. Proto je vhodné zvlášť zkoumat úspěšnost sítě na menším vzorku.
- Některé houby jsou od sebe objektivně nerozpoznatelné. Odborníci jejich příslušnost do jednotlivých druhů určují pomocí DNA.
- Vyhodnocení zcela opomíjí velmi důležitě dělení hub na jedlé/nejedlé případně jedlé-/nejedlé/jedovaté
- Testovacím vstupem je jediná fotografie houby. V praktickém použití máme však vždy k dispozici libovolné množství fotografií jednoho exempláře houby. Mnoho druhů hub však roste ve skupinách a máme tedy fotografie více exemplářů.

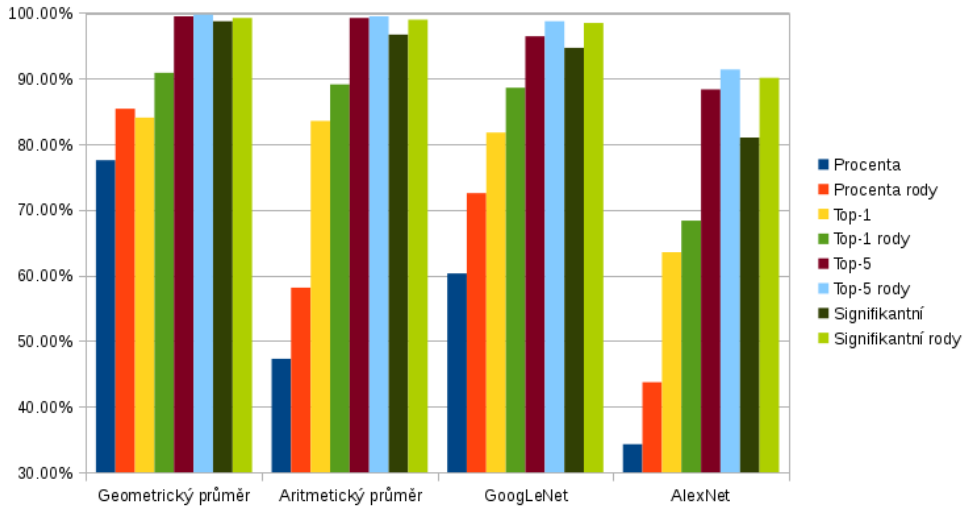
  Dále se proto pokusím tato fakta reflektovat a testovat klasifikátory tak, jak to bude nejlépe odpovídat jejich použití.

8.2 Vyhodnocení nejdůležitějších druhů v České Republice

Vytvořil jsem rozsáhlý dataset, který obsahuje velké množství různých druhů hub, kdy se pouze některé vyskytují na území české republiky a pouze některé druhy obsahují dostatečný

počet vzorků na trénování[1]. Prioritou mé aplikace ale je dosáhnout co nejlepších výsledků v houbách, které se ve velkém množství vyskytují na území České republiky. Pro výběr těchto hub jsem nenalezl seznam hub seřazený podle jejich výskytu v České Republice, takže jsem sám vytvořil seznam, který obsahuje 58 druhů hub, které se nacházejí zároveň v několika internetových atlasech hub. Lze se domnívat, že se jedná o houby, které jsou buď velmi rozšířené, nebo jsou pro houbaře jiným způsobem zajímavé, pokud se o nich rozhodlo psát několik na sobě nezávislých autorů.

Z vybraných 13 653 fotografií určených k testování jich pouze 967 patří do některého z uvedených druhů. Pro směrodatnější testování by bylo vhodné mít testovacích obrázků větší množství. Všechny ostatní fotografie byly ale použity ke trénování sítě a nejsou tedy k testování použitelné.

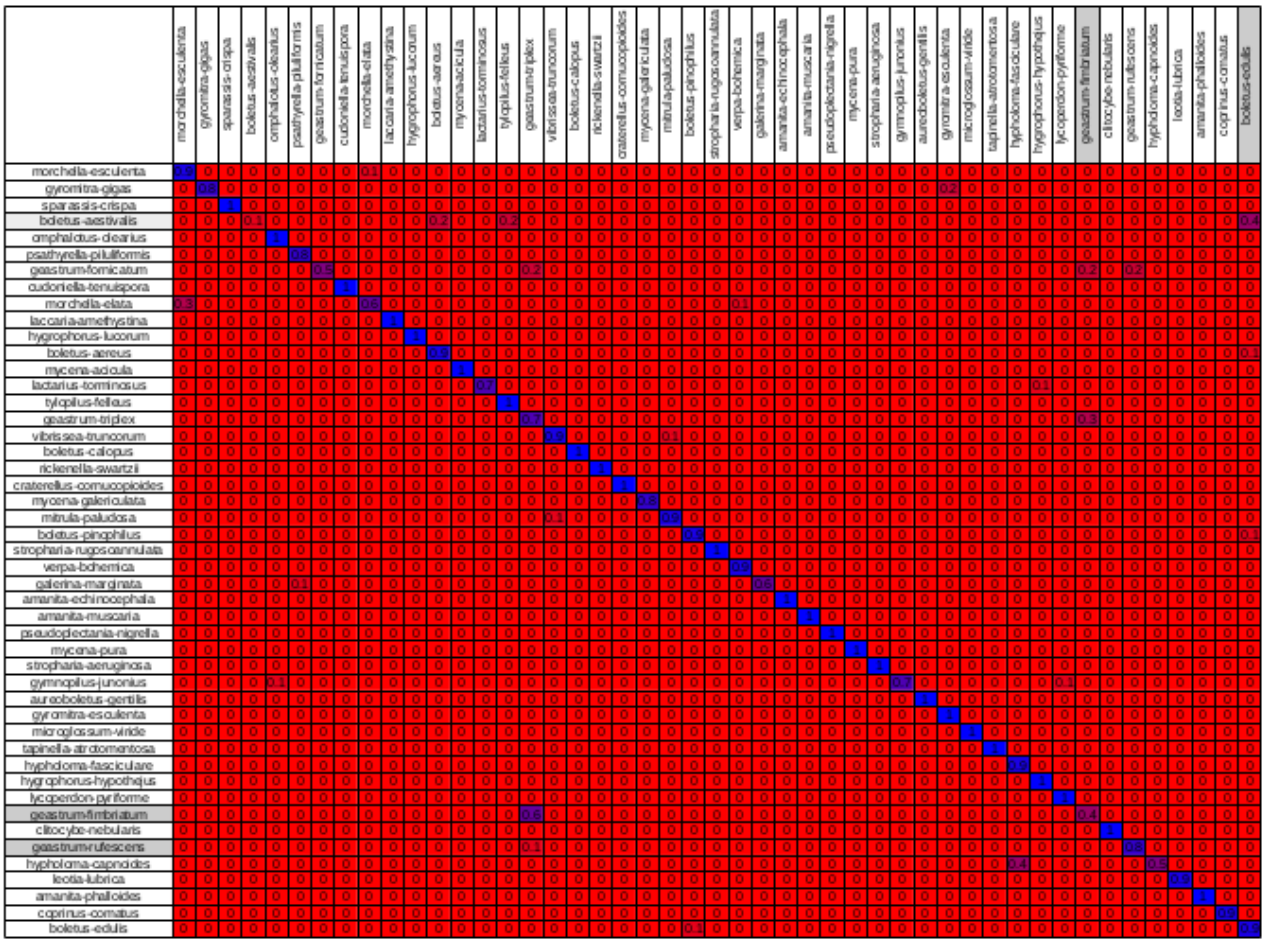


Obrázek 8.2: Vyhodnocení úspěšnosti klasifikace na vybraných druzích hub, které se nejčastěji vyskytují v České Republice

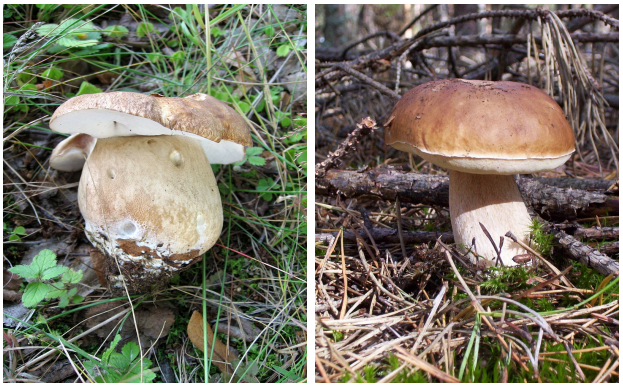
Výsledky znázorňuje graf 8.2. Výsledky jsou výrazně lepší než v případě testování nad celou datovou sadou. Je to mimo jiné způsobeno tím, že v těchto kategoriích obsahuje datová sada větší množství fotografií ke trénování sítí. Síť GoogLeNet dosáhla opět lepších výsledků než síť AlexNet. Nejlepších výsledků se však dosáhlo kombinací výstupů obou sítí. Jak aritmetický tak geometrický průměr dosáhly lepších výsledků než libovolná síť samostatně. Nejlepších výsledků se dosáhne výpočtem geometrického průměru obou sítí a následnou normalizací.

V tabulce změn 8.3 jsou vidět časté záměny Hřibu Smrkového(Boletus Edulis) a Hřibu dubového(Boletus Aestivalis). Odpovídá to i vizuální podobnosti obou druhů ukázané na obrázku 8.4.

[1]Jak jsem už zmínil v kapitole (6.1.3) houby se skutečně malým množstvím dat a nepoužitelné houby jsem odstranil



Obrázek 8.3: Tabulka záměn klasifikace pro nejčastější houby, které se vyskytují v ČR. Je zde například vidět častá záměna Hřibu Smrkového(Boletus Edulis) a Hřibu dubového(Boletus Aestivalis).



Obrázek 8.4: Houby, k jejichž záměně 8.3 často dochází Hřib Smrkový a Hřib Dubový. Fotografie Dezidor, Kami [13, 21]

	Jedlé	Nejedlé	Jedovaté
Jedlé	**95,72%**	3,74%	0,53%
Nejedlé	0,39%	**99,03%**	0,58%
Jedovaté	6,8%	2,91%	**90,29%**

Tabulka 8.1: Tabulka záměn hub z hlediska požívatelnosti

8.3 Zohlednění vzájemně nerozpoznatelných druhů

Existují houby, které jsou z vnějšího hlediska nerozpoznatelné [35], jak je vidět na obrázku 8.5

Obrázek 8.5: Těžko rozlišitelné druhy Boletus Smithii a Boletus Moravicus ,fotografie toroplanet.com

Bylo by tedy správné nepenalizovat klasifikátor za to, že neodlišil druhy, které pouze z fotografie odlišit nelze. Nepodařilo se mi však sehnat vhodná data a ruční vybírání shodně vypadajících druhů by bylo subjektivní a neúměrně náročné. Daná vlastnost nebyla tedy při testování zohledněna. Houby, které jsou často zaměňovány lze ale najít například v tabulce záměn 8.3.

8.4 Vyhodnocení z hlediska požívatelnosti

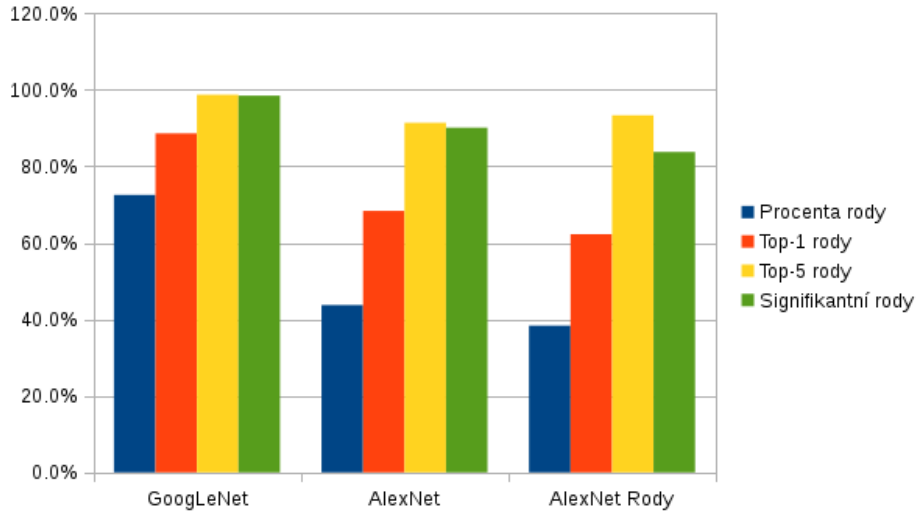
Zcela zásadní je z uživatelského hlediska dělení hub na jedlé a nejedlé. U mnoha hub v celé datové sadě se však toto hledisko vůbec nezohledňuje, proto jsem hodnocení úspěšnosti z hlediska požívatelnosti testoval opět nad nejčastějšími druhy České republiky.

V použitých internetových zdrojích se lze setkat s dělením podle následujících schémat (jedlé, nejedlé), (jedlé, nejedlé, jedovaté), (jedlé, nejedlé, jedovaté, smrtelně jedovaté). Nejčastěji se používá druhé, proto se ho budu držet i já.

Výsledky prezentuje tabulka záměn 8.1. Je z ní mimo jiné patrné, že v případě konzumace hub, které by klasifikátor označil za jedlé, je pravděpodobnost otravy pouze 0,53% na houbu.

8.5 Vyhodnocení Klasifikace na rody

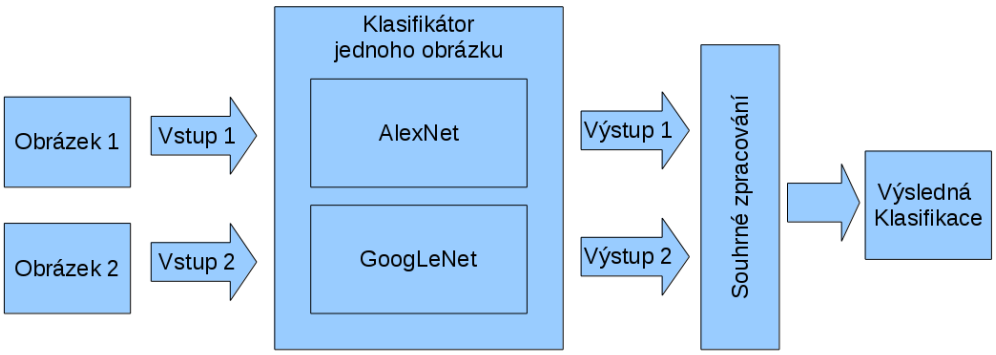
V sekci 5.2 byl popsán postup učení AlexNet sítě, která klasifikuje pouze s přesností na rody. Její výsledky jsem srovnal s výsledky sítí, které klasifikují s přesností na druhy a pravděpodobnost rodu se získá součtem pravděpodobností druhů, kteří do daného rodu patří.



Obrázek 8.6: Srovnání výsledků sítí klasifikující s přesností na druhy se sítí klasifikující s přesností na druhy

Graf 8.6 ukazuje, že trénování sítě pouze na úroveň nemá významný přínos. Výsledky jsou paradoxně horší, než výsledky sítí, které klasifikují na více kategorií.

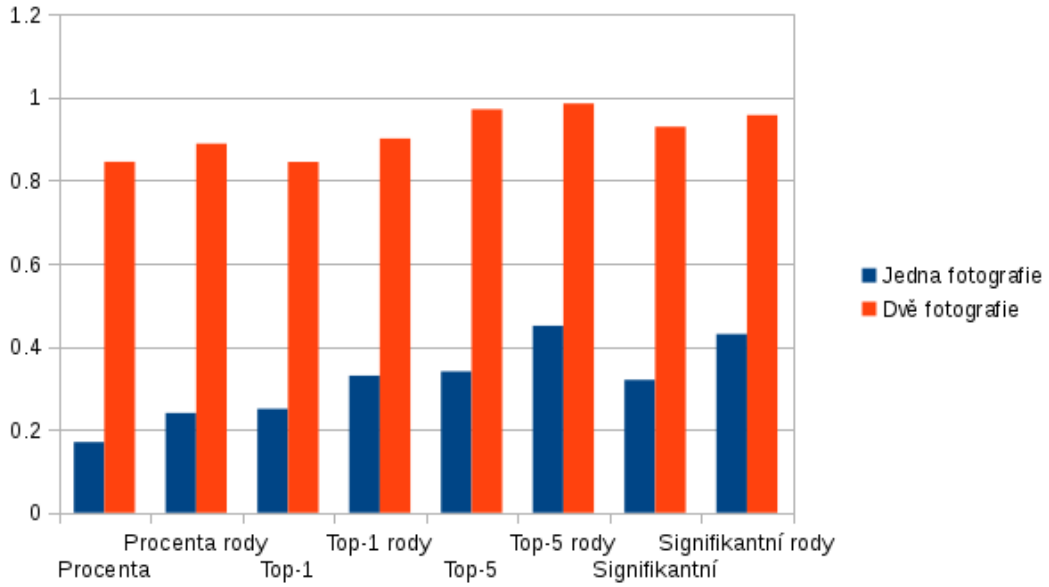
8.6 Klasifikace z více fotografií



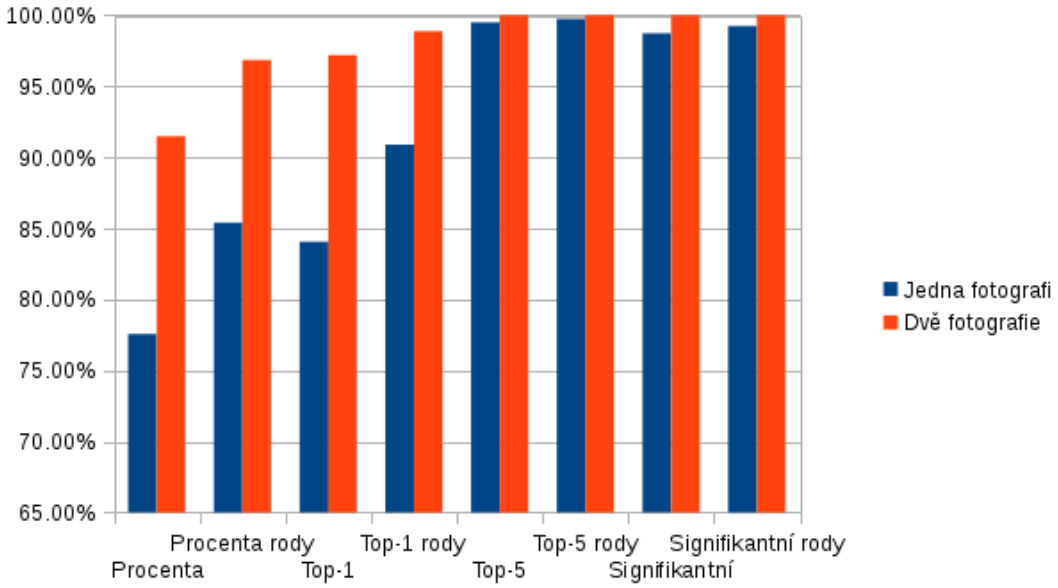
Obrázek 8.7: Schéma postupu klasifikace pro dvě fotografie jedné houby. Použitím dvou fotografií se výrazně zvyšuje úspěšnost klasifikace.

Pro účely mé aplikace není vhodné hodnocení na základě jedné fotografie houby. Z jednoho exempláře houby lze totiž vytvořit více různých fotografií. Získané výstupy neuronové sítě lze potom vhodně zkombinovat a dosáhnout tak vyšší úspěšnosti než při klasifikaci z jedné

fotografie [42]. Bohužel se mi nepodařilo získat ideální data k testování, tedy několik fotografií jednoho exempláře houby. Situaci lze napodobit tak, že budou klasifikovány dvě fotografie stejné houby a předpovědi se vhodně zkombinuíi. Bude to přesně simulovat velmi častou situaci, kdy houbař nalezne několik exemplářů stejného druhu. Pro skládání vektorů pravděpodobností byl součin a výsledek byl následně normalizován. Postup je naznačen na obrázku 8.7.



Obrázek 8.8: Srovnání výsledků klasifikace z jednoho obrázku vůči klasifikace ze dvou obrázků na celé datové sadě



Obrázek 8.9: Srovnání výsledků klasifikace z jednoho obrázku vůči klasifikace ze dvou obrázků na vybraných druzích v ČR

Předpovědi jsou velmi přesné, jak ukazují grafy 8.9 a 8.8. Na prvním grafu, který ukazuje výsledky testu pro všech 5097 druhů, je vidět zvýšení top-1 úspěšnosti z 23% na 82%. Druhý graf ukazuje, že mezi nejčastější druhy v ČR se úspěšnost zvýšila z 84% na 96%.

Kapitola 9

Závěr

V práci jsem popsal použití neuronových sítí pro klasifikaci obrazu. Nastudoval jsem metody zpracování obrazu, především ty, které jsou pro můj projekt použitelné. Experimentoval jsem s metodami na lokalizaci objektů a segmentaci obrazu a vybral jsem ty, které se ukázali být pro můj úkol použitelné, což byly konvoluční a plně konvoluční neuronové sítě. Dále jsem upravil a natrénoval několik neuronových sítí, které řeší jak problém detekce houby, tak problém klasifikace.

Výsledkem je použitelný program, který dává houbařům dodá základní orientaci, v okamžiku, kdy si vůbec nebudou jisti, do jakého rodu či druhu houbu zařadit. Uživatel si poté ověří správnost klasifikace v atlase hub. Očekávané využití spočívá v to, že poté co je na vstup přivedeno několik fotografií jednoho exempláře dané houby z různých stran se výsledky agregují a uživateli bude předloženo několik odhadů, o kterou houbu by se mohlo jednat. V úvahu přichází dát uživateli pět nebo deset možností, případně počet omezit tím, kolik odhadů vygenerovaných programem lze považovat za relevantní, kdy za relevantní označíme odhad, kterému bylo přiřknuto alespoň 5% pravděpodobnost.

Aplikace klasifikuje houby do 5097 různých druhů s top-1 úspěšností 23% respektive 35% s top-5 úspěšností. Výrazně vyšší úspěšnost ale dosahuje na houbách, které se v ČR vyskytují nejčastěji, kdy dosahuje 90 % top-1 úspěšnost respektive 99% top-5 úspěšnost.

Byl vytvořen plakát a video prezentující řešení diplomové práce. Jsou přiloženy na odevzdaném CD.

Práce mi pomohla hlouběji pochopit fungování neuronových sítí a vyzkoušet jejich nasazení na reálný problém.

9.1 Možnosti dalšího rozšíření

Ačkoliv jsem splnil všechny body zadání, zvolené téma jsem zdaleka nevyčerpal. Lepších výsledků lze dosáhnout rozšířením datové sady. Je k tomu možné využít množství veřejných databází a zdrojů na internetu. Seznam zdrojů je dostupný v přílohách.

Je možné zkoumat optimální velikost výřezů pro lokalizaci houby a vyzkoušet kombinovat výsledky jednotlivých metod lokalizace.

Dalším úkolem je čištění datové sady, protože zdroje odkud jsem čerpal obsahují chybně klasifikované obrázky, které potom samozřejmě snižují úspěšnost klasifikace.

Dále je možné využít faktu, že dostupné fotografie jsou většinou velmi kvalitní a experimentovat s klasifikací pomocí textury, kdy se trénovací data místo zmenšování oříznou a místo globálního pohledu na houbu bude k dispozici pouze detail s velkým rozlišením.

Dále pracuji na interaktivní aplikaci pro platformu android spojenou s atlasem hub, která bude komunikovat se serverem, kde budou vytvořené klasifikátory použity v praxi.

Literatura

[1] *Nearest Neighbor: Pattern Classification Techniques (Nn Norms : Nn Pattern Classification Techniques)*. Ieee Computer Society, 1990, ISBN 0818689307.
URL https://www.amazon.com/Nearest-Neighbor-Pattern-Classification-Techniques/dp/0818689307?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0818689307

[2] Houbaření. may 2017.
URL houbareni.cz

[3] Object Detection in Images. apr 2017.
URL https://www.saagie.com/blog/object-detection-part1

[4] Alex Krizhevsky, G. E. H., Ilya Sutskever: ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS 2012*, 2012.

[5] Association, I. M.: Species Fungorum. jan 2017.
URL http://www.speciesfungorum.org/

[6] Baranovič, J.: Nahuby.sk.
URL http://www.nahuby.cz

[7] Bazaraa M.S., J. J.: *Linear programming and network flows*. Georgia Institute of Technology, 1977, ISBN ISBN 0-471-06015-1.

[8] Bengio, Y.: Deep learning of representations for unsupervised and transfer learning. *Unsupervised and Transfer Learning Challenges in Machine Learning*, ročník 7, 2012: str. 19.

[9] Bishop, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007, ISBN 0387310738.
URL https://www.amazon.com/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0387310738

[10] Bordes, F.: GoogleNet. may 2017.
URL https://florianbordes.wordpress.com/

[11] Carrillo, H.; Brodersen, K. H.; Castellanos, J. A.: Probabilistic Performance Evaluation for Multiclass Classification Using the Posterior Balanced Accuracy. In *ROBOT (1)*, 2013, s. 347–361.

[12] Chu, H.: Mdb: A memory-mapped database and backend for openldap. 2011.

[13] Dezidor: Wikipedista Dezidor. may 2017.
URL https://cs.wikipedia.org/wiki/Wikipedista:Dezidor

[14] Duban, M.: *Identifikace osob pomocí hlubokých neuronových sítí*. diplomová práce, FIT VUT v Brně, 2016.

[15] Everingham, M.; Gool, L.; Williams, C. K.; aj.: The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision*, ročník 88, č. 2, Červen 2010: s. 303–338, ISSN 0920-5691, doi:10.1007/s11263-009-0275-4.
URL http://dx.doi.org/10.1007/s11263-009-0275-4

[16] Grayson Leonard, M. S.: Classifying edibility of mushrooms. Technická zpráva, University of California, 2012.

[17] Gunawardana, A.; Shani, G.: A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *J. Mach. Learn. Res.*, ročník 10, Prosinec 2009: s. 2935–2962, ISSN 1532-4435.
URL http://dl.acm.org/citation.cfm?id=1577069.1755883

[18] He, e. a., Kaiming: A global sampling method for alpha matting. In *CVPR 2011*, 2011, str. 2049–2056.

[19] Jens H. Petersen, D., Thomas Læssøe University of Copenhagen: Mycokey. jan 2017.
URL http://www.mycokey.com/

[20] Jia, Y.; Shelhamer, E.; Donahue, J.; aj.: Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[21] Kami: Wikipedista Kami. may 2017.
URL https://commons.wikimedia.org/wiki/File:Boletus_edulis_by_kami.jpg

[22] Kantorovich, L. V.: *Functional Analysis*. Pergamon Pr, 1982, ISBN 0080230369.
URL https://www.amazon.com/Functional-Analysis-Leonid-Vitalevich-Kantorovich/dp/0080230369?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0080230369

[23] Karpathy, A.: Convolutional Neural Networks for Visual Recognition. jan 2017.
URL http://cs231n.github.io/

[24] Kirk, D.: NVIDIA CUDA software and GPU parallel computing architecture. *ISMM*, ročník 7, 2007: s. 103–104.

[25] Lampert, C. H.; Blaschko, M. B.; Hofmann, T.: Beyond sliding windows: Object localization by efficient subwindow search. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, June 2008, ISSN 1063-6919, s. 1–8, doi:10.1109/CVPR.2008.4587586.

[26] LeCun, Y.; aj.: LeNet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 2015.

[27] Lichman, M.: UCI Machine Learning Repository. 2013.
URL http://archive.ics.uci.edu/ml

[28] Long, J.; Shelhamer, E.; Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. *CVPR (to appear)*, Listopad 2015, 1411.4038.

[29] Maxime Oquab, I. L. J. S., Léon Bottou: *Is object localization for free? - Weakly-supervised learning with convolutional neural networks*. 1996.

[30] Mcculloch, W. S.; Pitts, W.: A Logical Calculus of the Ideas Immanent in Nervous Activity*. *Bulletin of Mathematical Biology*, ročník 5, 1943: s. 115–133.

[31] Munakata, Toshinori: *Fundamentals of the New Artificial Intelligence*. Springer-Verlag New York, 2008, ISBN 978-1-84628-839-5.

[32] Nilsback, M.-E.; Zisserman, A.: Automated Flower Classification over a Large Number of Classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[33] Nilsback, M.-E.; Zisserman, A.: Automated Flower Classification over a Large Number of Classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[34] Prechelt, L.: A quantitative study of experimental evaluations of neural network learning algorithms: Current research practice. *Neural Networks*, ročník 9, č. 3, 1996: s. 457 – 462, ISSN 0893-6080, doi:http://doi.org/10.1016/0893-6080(95)00123-9.
URL http://www.sciencedirect.com/science/article/pii/0893608095001239

[35] Raja, H. A.; Baker, T. R.; Little, J. G.; aj.: {DNA} barcoding for identification of consumer-relevant mushrooms: A partial solution for product certification? *Food Chemistry*, ročník 214, 2017: s. 383 – 392, ISSN 0308-8146, doi:http://doi.org/10.1016/j.foodchem.2016.07.052.
URL http://www.sciencedirect.com/science/article/pii/S0308814616310743

[36] Savarese, S.: Computer Vision, Evaluation metrics and Caffe. apr 2017.
URL https://web.stanford.edu/class/cs231a/

[37] Smistad, E.: Some useful python functions for visualizing a caffe network. apr 2017.
URL https://github.com/smistad/visualize-caffe

[38] Szegedy, C.; Liu, W.; Jia, Y.; aj.: Going Deeper with Convolutions. *CoRR*, ročník abs/1409.4842, 2014.
URL http://arxiv.org/abs/1409.4842

[39] team, O.: OpenCV library. 2017.
URL http://opencv.org/

[40] Varda, K.: Protocol Buffers. http://code.google.com/apis/protocolbuffers/.

[41] Viola, P.; Jones, M.: Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, ročník 1, 2001, ISSN 1063-6919, s. I–511–I–518 vol.1, doi:10.1109/CVPR.2001.990517.

[42] Wei, W.; Barnard, E.; Fanty, M.: Improved probability estimation with neural network models. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, ročník 1, IEEE, 1996, s. 502–505.

[43] Whaley, R. C.; Dongarra, J. J.: Automatically Tuned Linear Algebra Software. In *Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*, SC '98, Washington, DC, USA: IEEE Computer Society, 1998, ISBN 0-89791-984-X, s. 1–27. URL http://dl.acm.org/citation.cfm?id=509058.509096

[44] Yosinski, J.; Clune, J.; Nguyen, A.; aj.: Understanding Neural Networks Through Deep Visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

[45] Yosinski, J.; Clune, J.; Nguyen, A.; aj.: Deep Visualization Toolbox. may 2017. URL https://github.com/yosinski/deep-visualization-toolbox

[46] Zeiler, R., M. D.; Fergus: *Visualizing and understanding convolutional networks.* 2014, 818–833 s.

[47] Zicha, O.: BioLib - Taxonomic tree of plants and animals with photos. jan 2017. URL http://www.biolib.cz

[48] Šimon, M.: *Rozpoznání objektů v obraze na platformě android.* bakalářská práce, FIT VUT v Brně, 2013.