Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Disertační práce:

Metody umělé inteligence v rozpoznávání rostlin

Autor: Petr Hanzlík

Vedoucí práce: doc. Ing. Arnošt Veselý, CSc.

Děkuji svém školiteli doc. Ing. Arnoštu Veselému CSc. za vedení této práce.

Abstrakt

Tato práce shrnuje poznatky v oblasti rozpoznání rostlin pomocí příznakových vlastností a metod umělé inteligence. Je vysvětlena role rozpoznání rostlin v precizním zemědělství. V práci jsou popsány a vysvětleny jednotlivé fáze obecného procesu rozpoznání objektů, včetně snímání obrazu, předzpracování, segmentace, extrakce příznaků, a klasifikace. Speciální pozornost je věnována konvolučním neuronovým sítím.

Vybrané metody rozpoznání rostlin založené na extrakci příznaků z obrazu jsou experimentálně aplikované na plevelné rostliny v raných fázích jejich růstu. Výsledky jsou srovnány s řešeními využívající konvoluční neuronové sítě. Hlavním výsledkem práce je metodika pro rozpoznání plevelných rostlin v kulturní vegetaci, kterou lze zobecnit i na další úlohy rozpoznání rostlin.

Klíčová slova:

rozpoznání rostlin, rozpoznání plevele, zpracování obrazu, extrakce příznaků, umělá inteligence, konvoluční neuronové sítě

Abstract

This work summarizes knowledge in the field of feature-based plant recognition and methods of artificial intelligence. The role of plant recognition in precision agriculture is explained. The individual stages of general object recognition process are reviewed, including image acquiring, pre-processing, segmentation, feature extraction, and classification. Special attention is paid towards methods convolutional neural networks.

Several feature extraction-based methods are experimentally applied on weed plants in the early stages of their growth. The results are compared with solutions that take advantage of convolutional neural networks. The principal output of this thesis is a methodical framework for weed plant recognition, which can be further generalized to other areas of plant recognition research.

Keywords

plant recognition, weed recognition, image processing, feature extraction, artificial intelligence, convolutional neural networks

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Výskyt nežádoucích rostlin v kulturní vegetaci má významné negativní dopady na celý proces zemědělské výroby. Oerke (2006) odhaduje, že celosvětové ztráty způsobené všemi škůdci snižují rostlinnou výrobu o 40% až 80%, v závislosti na druhu pěstované plodiny a místních podmínkách. Zároveň pak uvádí, že nejvýznamnějším zdrojem těchto ztrát jsou právě plevelné rostliny. Současná zemědělská výroba tak musí souvisle bojovat s nežádoucím výskytem plevelných rostlin v zemědělské krajině.

Metodiky hubení (popř. regulace) plevele v zemědělské krajině mohou být obecně klasifikovány jako preventivní, biologické, fyzikální a chemické. V konvenčním pojetí řízení rostlinné výroby je nejrozšířenější metodou plošné použití herbicidních látek (Ghersa, et al., 2000), (Liphadzi & Dille, 2006), tedy chemické hubení plevele. Tato metoda je účinná a relativně levná, zejména pak v porovnání s metodami jako je manuální mechanické odstraňování plevelných rostlin. Nevýhodou jsou však negativní účinky herbicidů na životní prostředí, jak dokládají např. (Geigera, et al., 2010), (Aktar, et al., 2009) či (Relyea, 2005). Jako možné řešení tohoto problému vznikly metodiky známé pod zkratkou IPM (Integrated pest management), jejichž cílem je minimalizovat dopady chemických látek použitím vhodné kombinace existujících metodik hubení plevele při dosažení obdobných výnosů jako v konvenčním pojetí (Radclife, et al., 2009).

Výše zmíněné techniky IPM jsou založeny na rovnoměrném, plošném ošetřování zemědělské půdy. Plevele se však v polích nevyskytují rovnoměrně, ale jsou seskupeny v lokálních ohniscích (Dieleman & Mortensen, 1999), (Gerhards & Christensen, 2003). Tohoto faktu využívá skupina metodik a agronomických postupů, obecně označovaných zastřešujícím pojmem *precizní zemědělství*. V precizním zemědělství jsou problémové oblasti spravovány a ošetřovány na lokální bázi v závislosti na rozličných typech půdy, charakteru krajiny, pěstované plodině, či detekovaném plevelu a jeho ohniscích v polním porostu. Tyto postupy obvykle vedou ke zvýšené ziskovosti díky optimalizaci použitých vstupů do zemědělské výroby (Larson & Robert, 1991), (Zhang, et al., 2002) jakož i ke zlepšení kvality životního prostředí (Mulla, et al., 2002) (Mulla, 2013) (Tian, 2002). De Castro uvádí, že automatická identifikace plevelů může snížit spotřebu herbicidních látek o 4-94% (de Castro, et al., 2012), (de Castro &

López-Granados, 2013) v závislosti na typu plevele a míře jeho rozšíření. Důležitým poznatkem je také fakt, že hustota plevele v kulturní vegetaci dlouhodobě nestoupá, je-li plevel regulován pouze v lokálních ohniscích, jak prokázali ve své dlouhodobé studii Ritter a Gergards (2007). Tuto strategii hubení plevele lze tedy považovat za využitelnou v praxi. Na druhou stranu je také nutné zmínit, že ekonomická výhodnost takového řešení je závislá na úrovni zamoření plevely. Andújar, et al. prokázali v ekonomickém srovnání, že při výskytu plevele na více než 76% plochy je jednoznačně výhodnější varianta plošného postřiku (Andújar, et al., 2011).

Jedním ze základních předpokladů aplikace herbicidních látek na lokální úrovni je detailní přehled druhového složení plevelných rostlin v pozorované oblasti spolu s hustotou jejich výskytu a rozmístění hlavních ohnisek (Wiles, 2005). Tuto informaci lze získat manuálním sčítáním výskytů plevelů na obhospodařované ploše, což je však náročné na čas i lidské zdroje, a tudíž drahé, zejména pak pro velké polní oblasti. Jako řešení se nabízí systém počítačového vidění schopný rozpoznat plevelné rostliny, vyhodnotit míru jejich rozšíření na poli a zanést jejich polohu do mapy.

Pro realizaci takových systémů je nutné rostlinu detekovat a přesně klasifikovat na základě jejích vizuálních charakteristik. Robustní automatické řešení schopné detekce (a rozpoznání) plevelů však v současnosti chybí, což Slaughter et al. (2008) označili jako hlavní překážku při tvorbě autonomních robotických řešení pro hubení plevelů. Tato úloha je značně problematická vzhledem k možnému výskytu velkého množství různých druhů plevelů, jejichž vzhled se mění během jednotlivých fází jejich vegetativního růstu.

Tento výzkum vychází z hypotézy, že je možné najít dostatečně stabilní parametry, které mohou být použity pro rozpoznání jednotlivých druhů plevelných rostlin. Tyto parametry využívají tvarových vlastností, barvy a textury pozorovaného objektu, popřípadě vzájemných vztahů objektů v obrazu. Vzhledem k značné rozmanitosti úlohy jsou poté rostliny obvykle rozpoznány na základě těchto vlastností pomocí algoritmů strojového učení schopných generalizace.

2. Cíle práce

Správa výskytu plevelných rostlin na místní bázi vyžaduje získání detailního přehledu o vyskytujících se plevelných rostlinách a četnosti jejich rozšíření na jednotlivých částech pozemku. Pro každou pěstovanou polní plodinu je nutné posuzovat mnoho druhů plevelných rostlin, zejména však takové, které způsobují vysoké ekonomické ztráty. Expert posuzuje výskyt plevelných rostlin v předem určené oblasti (např. 0,25 m^2) na základě jejich vnějších znaků, zejména pak morfologických vlastností, tedy tvaru a celkového vzhledu jednotlivých rostlin, jakož i dalších specifických vlastností (žilnatina listu, přítomnost trichomů, aj.). Následně pak ve vybrané oblasti spočte absolutní počet plevelných rostlin, nebo odhaduje, jaký je procentuální poměr plevele k pěstované rostlině. Na základě takovéhoto expertního posouzení jsou potom provedeny agronomické zákroky, které mají za cíl výskyt plevele omezit. Chybějícím článkem pro automatizaci takovéhoto procesu je pak chybějící řešení schopné detekce a rozpoznání plevelných rostlin (Slaughter, et al., 2008).

Hlavním cílem této práce je nalezení vhodného řešení pro rozpoznání plevelných rostlin v raných fázích jejich vegetativního růstu (ohraničeno dle definice Forbese a Watsona (1992)), pomocí metod obrazové analýzy a metod strojového učení. Specifickým cílem je rozpoznání plevelných rostlin v rané růstové fázi, kdy se rostlina vyznačuje pouze dvěma děložními lístky (*cotyledony*), popřípadě dvěma páry děložních lístků. Pro dosažení tohoto cíle je nutné vyřešit následující dílčí cíle:

- Nalezení kombinace metody snímání obrazu a předzpracování obrazu pro účinnou segmentaci plevelných rostlin a půdy.
- Vytvoření výchozí databáze klasifikovaných obrazů pro další rozpoznání plevelných rostlin a polních plodin a jejich anotace ve snímcích.
- Stanovení příznakových vlastností objektů a posouzení jejich vhodnosti pro rozpoznání plevelných rostlin v raných fázích jejich vegetativního vývoje.
- Klasifikace plevelných rostlin prostřednictvím metod strojového učení na základě vybraných vlastností.

Tento výzkum je dílčím krokem ve vývoji autonomního systému pro detekci, rozpoznání a konečného řízení výskytu plevelných rostlin v kulturní vegetaci v systémech precizního zemědělství.

3. Metodika

Tato práce se zabývá metodikami zpracování obrazu, příznakovým popisem objektů v něm se nacházejících a následnou klasifikací nalezených tvarů pomocí metod umělé inteligence. Výsledky této práce jsou tedy závislé na čtyřech hlavních krocích:

- Snímání obrazu.
- Předzpracování a segmentace obrazu.
- Popis objektů a jejich vlastností.
- Klasifikace objektů.

Výběr vhodné metody snímání obrazu bude proveden na základě analýzy a syntézy poznatků z odborné literatury a posouzení vhodnosti výsledného obrazu pro další zpracování. Snímání obrazu bude s ohledem na algoritmické zpracování pro aplikace strojového učení provedeno ručně ze stabilní vzdálenosti tak, že výsledný obraz bude obsahovat plevelnou rostlinu co nejblíže svému středu. Vzhledem k velké možné variabilitě úlohy související s osvětlením jsou v momentě snímání předpokládány stabilizované podmínky osvětlení. V každém obraze se může vyskytovat vícero rostlin jak jednoho, tak i více druhů, i další objekty přírodního původu (kameny, suchá stébla aj.). Každý obraz bude následně expertně klasifikován na základě druhové příslušnosti centrálního objektu. Výsledkem tohoto kroku tedy bude reprezentativní množina obrazů vybraných plevelných rostlin.

Na základě metodiky získávání obrazu bude zvolena strategie předzpracování a následné segmentace obrazu. Předpokládá se, že základní segmentační problém stanovené úlohy, tedy rozlišení rostlin a půdy, lze řešit použitím specifických vegetačních indexů, a následným prahováním. Bude také posouzena možnost rozlišení jednotlivých částí vegetace na základě barvy a textury. Výsledkem segmentace je rozdělení vstupní obrazové funkce $f(x, y)$ na n regionů popředí (rostliny) a pozadí. Obraz je následně očištěn od objektů malé velikosti a jednotlivé celistvé regiony označeny.

Takto nalezené objekty jsou popsány pomocí vybraných obrazových příznaků, včetně jednoduchých tvarových deskriptorů, momentových invariantů, metod založených na transformaci objektu do polárních souřadnic vycházejících z jeho těžiště a lokálních binárních

vzorů pro popis textury. Jsou popsány také metody, které jsou vhodné pro popis specifických částí listů, včetně žilnatiny a hrany čepele listu. V práci bude také řešena úloha nalezení normalizované pozice objektu vhodné pro přímou klasifikaci.

Takto vybrané objekty nebo jejich specifické příznakové vlastnosti budou použity jako vstup pro klasifikační metody. Krom tradičních přístupů sestávajících z řetězce snímání-předzpracování-extrakce příznaků-klasifikace jsou v práci využity také různé varianty konvolučních neuronových sítí, které provádějí předzpracování a extrakci příznaků v rámci své architektury.

V rámci práce je vzhledem k předpokládané složitosti a rozmanitosti úlohy kladen důraz na použití metod strojového učení schopných generalizace. Pro aplikaci klasifikačních metod pak bude použito standardní rozdělení vstupní datové množiny na podmnožiny učební, validační a testovací. Vhodnost klasifikační metody bude posuzována na základě relativní přesnosti klasifikace na testovací množině, matici záměn a analýze citlivosti.

Na základě výše uvedených kroků pak bude vytvořena komplexní metodika pro rozpoznání plevelných rostlin v aplikacích precizního zemědělství.

4. Informační technologie v precizním zemědělství

Vznik a rozvoj precizního zemědělství je důsledkem technologického rozvoje, ke kterému v zemědělství došlo od devadesátých let. Tento rozvoj je ve velké míře spojený se zvýšenou dostupností dat vycházející z rozvoje informačních a komunikačních technologii, jakož i zvýšenou kvalitou a přesností senzorů, které mohou být upevněny na letadlech, bezpilotních vzdušných dopravních prostředcích (drony, kvadrokoptery, atd.) (Berni, et al., 2009) (Zhang & Kovacs, 2012) (Primicerio, et al., 2012), polní mechanizaci (Long, et al., 2008), (Adamchuk, et al., 2004), či mobilních robotech (Astrand & Baerveldt, 2002). Díky těmto technologiím je tak možné získat detailní přehled o stavu polí a plodin na nich pěstovaných. Současná úroveň dostupnosti dat vede k úvahám, že je možné popsat stav porostu natolik přesně, aby bylo možné rozlišit, a tedy spravovat jednotlivé rostliny.

Systémy precizního zemědělství se obecně skládají ze třech základních komponent (Weis, et al., 2008):

- Senzory, které jsou schopny identifikovat a změřit proměnlivost podmínek vegetace a prostředí (Godwin & Miller, 2010)
- Analytické a rozhodovací mechanismy, která slouží ke stanovení strategie obhospodařování pole (Kudsk, 2008)
- Technologie schopná variabilní aplikace agrotechnických postupů, v závislosti na zvolené strategii (Auernhammer, 2001)

Metodiky precizního zemědělství byly jednotlivě využity v celé řadě aplikačních oblastí. Následující přehled uvede odkazy na některá klíčová řešení v oblasti precizního zemědělství popsaná v literatuře, které mohou vézt k realizaci automatické správy polních vegetace:

Naváděcí systémy: přesné řízení na poli tak, aby nedocházelo k překryvu obhospodařovaných oblastí (Zier, et al., 2008)

Půdní hospodářství: obdělávání půdy (např. hloubka orby) v závislosti na vlastnostech půdy (Adamchuk, et al., 2004)

Hnojení: množství aplikovaného hnojiva odpovídá lokálnímu stavu půdy na poli (Biermacher, et al., 2009)

Ochrana rostlin: pesticidní látky (herbicidy, fungicidy, insekticidy) jsou aplikovány pouze tam, kde je to nutné (Miller, 2003)

Zavlažování: přesné zavlažování na základě měření vlhkosti půdy (Al-Kufaishi, et al., 2006)

Mapování výnosů: kontrola kvality manažerských rozhodnutí a lokální evidence výnosů (Arslan & Colvin, 2002)

Dokumentace: veškeré provedené operace mohou být přesně dokumentovány pro každou ze spravovaných zón na poli, včetně informací o celkovém množství použitých vstupů pro dosažení výnosů

Klíčovým předpokladem pro praktické použití většiny výše uvedených aplikaci precizního zemědělství je přesná znalost polohy geografické pozice. Většina současných systémů využívá přijímačů GPS (global positioning system) pro měření pozice v poli na základě satelitních signálů, případně doplněných jedním či více signály referenčních stanic (RTK – real time kinematic). Tato prostorová data jsou následně zanášena do geografických informačních systémů (GIS), s jejichž pomocí mohou být tato data ukládána, upravována a analyzována. Často používané jsou pak analýzy založené na kombinaci rozličných datových množin (vrstev) pomocí jejich vzájemného průniku a prostorové interpolace proměnných. Tato data a výsledky provedených analýz mohou být následně vizualizovány v podobě tematických map. V precizním zemědělství jsou GIS používány zejména pro tvorbu aplikačních map, na jejichž základě jsou na poli prováděny cílené agronomické zásahy. Geografické informační systémy jsou tedy nedílným předpokladem pro fungování naprosté většiny výše uvedených aplikací v precizním zemědělství.

5. Teoretické principy rozpoznání obrazu

Základním prostředkem pro automatizované rozpoznání plevelů a rostlin obecně jsou metody strojového vidění. Systémy pro rozpoznání plevelných rostlin se obecně skládají ze čtyř následujících funkčních bloků, které lze metodicky řešit i samostatně:

- Snímání obrazu
- Předzpracování a segmentace obrazu
- Popis objektů a jejich vlastností
- Klasifikace objektů

Následující kapitoly popíší některé principy těchto funkčních bloků a metody, které byly úspěšně použity při rozpoznání plevelů.

5.1. Rozpoznání rostlin (na základě listů) v literatuře

Tato práce se bude nadále zabývat zemědělskými systémy pro ochranu rostlin, konkrétně pak problematikou rozpoznání rostlin, která je jedním z nezbytných předpokladů pro efektivní agronomické zákroky v systémech precizního zemědělství (Slaughter, 2013).

Rozpoznání (klasifikace) obrazu je obecně definováno jako označení obrazu či vzoru pomocí předem definované kategorie (třídy). V případě rozpoznání plevelných rostlin je tedy cílem označit rostlinu rodovým a druhovým jménem. Expertní klasifikace založená na vizuálním rozpoznání plevelu je obecně založena na rozlišení vnějších charakteristik rostliny, zejména pak barvy, tvaru (včetně vzájemné pozice listů) a dalších specifických povrchových vlastností. Z dostupných přehledových studií zaměřujících se na počítačové viděné pak lze usuzovat, že základní rozlišovací prvky jsou ekvivalentní. Mezi hlavními znaky pro rozlišení mezi jednotlivými druhy plevelných rostlin převládá tvar (Slaughter, 2013), (Kluge & Nordmeyer, 2009) (Hemming, 2000), (Perez, et al., 2000) a v případě vhodných vlastností pěstované plodiny i barva, jak ukazují například Jafari et.al na případě cukrové řepy (Jafari, et al., 2006).

Plevele v běžných polních plodinách začínají snižovat výnosy již mezi čtvrtým a šestým týdnem od vyklíčení (Cousens, et al., 1987). Rozpoznání plevelů tedy musí být provedeno již v raných fázích jejich růstu. Zároveň pak platí, že doba klíčení je závislá na podmínkách, a ve

stejné době se tak v polním porostu může vyskytovat rostlinný druh v různých vegetativních fázích růstu. Úloha rozpoznání objektu přírodního původu (rostliny) v přírodních podmínkách je pak navíc komplikována celou řadou problémů. Mezi nejvýznamnější patří:

- Nerovnoměrné osvětlení v reálných podmínkách
- Vzájemný překryv jednotlivých rostlin
- Proměnlivý vzhled rostliny během růstu
- Značná rozmanitost vzhledu v rámci jednoho druhu i růstové fáze (částečně závisí na podmínkách v místě růstu)
- Možnost mechanického, chemického či biologické poškození

V důsledku těchto problémů algoritmy navržené ve stabilizovaných laboratorních podmínkách a řešení pracující s idealizovanou množinou často při praktických testech selhávají. Následující přehled popíše výsledky vybraných výzkumů, které se zabývaly rozpoznáním plevelů a rostlin obecně, a jejich klíčové metodické principy.

Rozpoznáním rostlin se zaměřením na aplikace v precizním zemědělství se zabývá celá řada studií. Některé z nich jsou založeny na rozlišení mezi pěstovanou plodinou a plevely, jiné se zaměřují na klasifikaci jednotlivých plevelných rostlin, které umožňuje cílené zásahy pouze v místech, kde se vyskytují plevele, které by mohly způsobit významné ekonomické škody.

Většina autorů ve svých studiích vychází z drobných zjednodušujících předpokladů, které umožňují prozkoumat specifické aspekty navrhovaných metod. Mezi značně zjednodušená řešení patří například metoda, kdy je popsána každá nalezená rostlina (resp. objekt) pomocí plochy (vyjádřené jako absolutní počet pixelů). Rozlišení mezi pěstovanou plodinou a plevelem je potom provedeno na základě pevně stanovené mezní hodnoty velikosti objektu (Hlaing & Khaing, 2014). Prakticky totožného přístupu využívá také (Siddiqi, et al., 2008). Ačkoliv je takovýto přístup velmi jednoduchý na implementaci, jeho praktické použití je možné pouze v případech, kdy je jedna z rostlin v pozorovaném porostu výrazně dominantní a jednotlivé rostliny jsou vzájemně odlišitelné.

Jeon et al. (2011) rozlišují mezi rostlinami kukuřice a plevely na základě pěti normalizovaných příznaků odvozených ze 4 základních morfologických vlastností rostlin (obvod, plocha,

maximální a minimální vzdálenost mezi dvěma pixely na hraně). Tyto příznaky jsou použity jako vstup pro dopřednou neuronovou síť. Autoři dosahují klasifikace kukuřičných rostlin vyšší než 90% (92,5% a 95,1% ve dvou provedených testovacích iteracích). Podobné metody využívají i Kiani a Jafari (2012), přičemž také porovnávají metody rozlišení plevele a plodiny pomocí diskriminační analýzy a neuronové sítě. Při použití neuronové sítě a 7 morfologických znaků (poměr nejdelší a nejkratší osy, kompaktnost, elongace, poměr obvodu a šířky, poměr délky a obvodu, aj.) dosahují přesnosti klasifikace plevelných rostlin i kukuřice až 100 %. Je ale nutné podotknout, že úloha je značně zjednodušena dominantní velikosti kukuřice. Jednoduché tvarové deskriptory využívají i Cho, et al. (2002) pro rozlišení ředkve a plevelů, přičemž dosahují přesnosti klasifikace 93,3 % pro ředkve a 93.8 % pro plevele při klasifikaci pomocí neuronové sítě.

Zajímavé metody extrakce příznaků lze nalézt také ve výzkumu zaměřujícím se na jednotlivé části rostlin. Lee a Hong navrhují metodu extrakce žilnatiny, která vychází v provedení několikanásobného morfologického otevření na intenzitním obrazu $f(x, y)$, který obsahuje právě jeden list. Výsledkem této operace je obraz $g(x, y)$. Žilnatina je potom vyjádřena jako rozdíl mezi obrazem $g(x, y)$ a původním obrazem $f(x, y)$ a následně popsána ve dvou rovinách pomocí Fourierovy frekvenční transformace (FFT) (Lee & Hong, 2013).

Většina výše popsaných postupů naneštěstí neposkytuje dostatečně detailní informaci pro zachycení jemných detailů, jako jsou zvlnění čepele listů. Tohoto problému si všímají Cerutti et al., kteří vycházejí ze základního polynomického modelu listu (Cerutti, et al., 2011) pomocí něhož je čepel listu (resp. její hrana) rozdělena na základnu, strany a vrchol, přičemž každá z těchto částí je následně popsána pomocí transformace do prostoru zakřivení (curvature-scale space transformation), která popisuje konkávní a konvexní charakter hrany v intervalech danými lokálními maximy a minimy (Cerutti, et al., 2014), tedy vrcholy a základnami jednotlivých „zoubků“ na hraně. Při použití této metody potom dosahují přesnosti klasifikace až 80% (Cerutti, et al., 2014).

V mnoha aplikacích byly zaznamenány úspěchy také při použití textur pro rozpoznání rostlin. Tang et al. (2003) vytvořili aplikaci, která je schopna rozlišit mezi širokolisté plevely a travinami na základě popisu textury pomocí filtrační masky založené na Gaborových vlnkách

(Gabor wavelets filter mask) o velikosti 17x17px ve čtyřech frekvenčních úrovních. Pro každou frekvenční úroveň je vypočten konvoluční výstup reálné (R_{j-ave}) a imaginární (I_{j-ave}) části jako $\sqrt{R_{j-ave}^2 + I_{j-ave}^2}$. Výsledkem je příznakový vektor o čtyřech složkách, který je použit jako vstup pro dopřednou neuronovou síť. Autoři na testovací množině čítající 20 obrázků vykazují úspěšnost rozlišení 100 % (Tang, et al., 2003). Analogického postupu využili také Chaki a Parekh (2012) pro rozlišení 3 druhů listů, přičemž i oni dosahují úspěšnosti 100 %. Qi et al. (2012) použili pro rozpoznání deskriptor textury označovaný jako *Pairwise Rotation Invariant Co-occurrence Local Binary Pattern* pomocí něhož dosáhli přesnosti 99.38 % na testovací množině 15 druhů stromů. Rozpoznání rostlin pomocí analýzy textury se tedy jeví jako velmi slibný přístup.

Pro rozpoznání listů stromů byly také úspěšně využity různé varianty konvolučních neuronových sítí. Jheon a Rhee (2017) využili architekturu konvoluční neuronové sítě GoogLeNet (také Inception) pro rozpoznání listů stromů z datové množiny Flavia sdružených do skupin dle tvaru (oválné, srdčité, složené atd.). Dále pak ověřovali schopnosti sítě rozlišit tyto rostliny v situacích, kde došlo k poškození barvy a textury (oschnutí) a mechanickému poškození, přičemž celková přesnost klasifikace se pohybuje v závislosti na míře poškození mezi 95-98%. Sun, et al. (2017) pak za použití hluboké architektury ResNet (varianta ResNet26) dosahují přesnosti klasifikace jednotlivých rostlinných druhů 91.78%. Zhu et. al (2018) využívají variantu architektury VGG pro klasifikaci rostlin na základě obrázků v přirozeném prostředí z databáze LifeCLEF2015, přičemž výstupní klasifikační MLP nahrazuje klasifikátor SVM. Při klasifikaci části databáze obsahující rostliny toto řešení dosahuje relativní přesnosti 67,10%. Vícevrstvé konvoluční architektury inspirované architekturou AlexNet, využili také Lee et al. (2015) při rozpoznání 44 druhů rostlin na základě listů shromážděných Královskou botanickou zahradou v Kew. V (Lee, et al., 2015) dosahují přesnosti klasifikace 99,5%. Ačkoliv některé z výše zmíněných výzkumů očividně pracují s poměrně jednoduchou množinou, konvoluční neuronové sítě lze v současnosti považovat za jednu z nejvhodnějších metod pro rozpoznání objektů v obrazu.

Prací, které se zabývají rozpoznáním rostlin na základě děložních lístků (lat. kotyledon) je poměrně malé množství, vzhledem ke komplikovanému charakteru úlohy, danému značnou

podobností rostlin v této růstové fázi. Kluge a Nordmeyer rozlišují mezi děložními lístky rozrazilu břečťanolistého (*Veronica hederifolia*) a svízele přítulné (*Galium aparine*) (Kluge & Nordmeyer, 2009), přičemž dosahují celkové přesnosti klasifikace 91 %. Každá nalezená rostlina je umístěna v normalizované pozici, tak, že bod, kde se listy spojují (autory označovaný jako balance point) je umístěn do počátku souřadnicové soustavy, a rostlina je následně natočena tak, že její hlavní osa je rovnoběžná s osou x. Tvar rostliny je potom popsán pomocí úhlů, které svírá úsečka daná bodem na hraně a počátkem souřadnicové soustavy s osou x. Pro každou rostlinu je popsán právě jeden list (je předpokládána symetrie) pomocí 4 spočtených úhlů, které jsou použity jako příznakový vektor, tedy jako vstup pro neuronovou síť. Tato metoda je v popsané podobě nezávislá na velikosti objektu, jeho pozici a rotaci, avšak nalezení výchozí normalizované pozice je popsané zjednodušeně. Tato metoda také neuvažuje se vzájemným přeryvem rostlin a možnými deformacemi jejich tvaru.

Existují také postupy, které namísto příznakového rozpoznání spoléhají na dostatečně velkou učební množinu, která postihuje variabilitu úlohy. Takovým typem řešení je rozpoznání obrazu normalizovaného na předem stanovený rozměr pomocí metod strojového učení, nejčastěji pak pomocí neuronových sítí. Yang et al. (Yang, et al., 2002) využili operace poloprahování založené na dominanci hodnot zeleného kanálu (přístup ekvivalentní k přístupu „Excessive green“) pro výběr rostlin. Hodnoty intenzity takto vzniklých objektů jsou následně převedeny do tónů šedi a normalizovány do uzavřeného intervalu $< 0; 1 >$. Každý z takto detekovaných objektů je následně vyříznut a přizpůsoben obrazu o velikosti 80x80px, který je použitý jako vstup pro dopřednou neuronovou síť. Autoři uvádějí míru rozpoznání cílové plodiny (kukuřice) až 100 %, přičemž schopnost rozpoznání plevelných druhů se pohybuje mezi 62-90%. Autoři i přes povzbudivé výsledky dospěli k závěru, že je takto možné řešit výskyt plevele pouze mimo stanovené výsevní řady.

5.2. Snímání obrazu

Tato přehledová studie, i následný výzkum s ní související se zabývá pouze metodami zpracování dvourozměrného obrazu, v nichž je obraz matematicky definován pomocí funkce dvou proměnných $f(x, y)$, jejíž funkční hodnoty nejčastěji odpovídají intenzitě zachyceného

signálu. Digitální obraz lze chápat jako průmět trojrozměrného prostoru na matici čísel popisující určitý pohled. Jako snímání obrazu je potom označováno zaznamenání určitého pásma elektromagnetického záření pomocí optického zařízení (fotografický aparát, kamera, spektrograf, spektrofotometr aj.) obsahujícího optický senzor. Funguje-li snímací zařízení analogově, je také nutné signál převést do diskrétní, digitální podoby.

Metody strojové vidění zpracovávající digitální obraz jsou v zemědělství využívány v řadě aplikací, včetně pozemního a dálkového průzkumu stavu porostu, jakož i při kontrole kvality zemědělských produktů či v automatizaci procesů. Tyto systémy využívají metody snímkování, které zaznamenávají zejména viditelné světlo (VIS), infračervené záření (IR) a ultrafialové záření (UV), ale i další spektra elektromagnetického záření. Zevrubný přehled těchto popisů vypracovali například Chen et al. (2002).

Systémy pro pořizování obrazových záznamů by měly v podmínkách precizního zemědělství fungovat i za proměnlivých podmínek osvětlení a nepříznivých klimatických podmínek. Tradiční digitální fotografie, která zaznamenává viditelnou oblast světla ve formátu RGB, je však na takovéto změny značně citlivá, zejména při aplikacích založených na analýze barvy (El-Faki Mozib, et al., 2000). Je tedy nutné použít speciální přípravky, které jsou schopny fotografovanou oblast přistínit nebo osvětlit tak, aby byly podmínky scény relativně stabilní.

5.2.1. Multispektrální a hyperspektrální snímkování

Možným řešením výše zmíněného problému je zaznamenat obraz ve spektrech, která nejsou na změny v momentálním přirozeném osvětlení (tedy ve viditelném světle) tak citlivé. K tomu slouží multispektrální snímkování, které spočívá v zaznamenání fotografované scény v několika přesně vymezených spektrálních pásmech zároveň. Spektrometrické metody mohou zaznamenávat odrazivosti objektů v pásmech viditelného světla, infračerveného či ultrafialového záření, jakož i přirozenou, či indukovanou fluorescenci (Pahikkala, et al., 2015).

V precizním zemědělství je potom velmi rozšířené multispektrální snímkování, které zaznamenává obraz v pásmech modrého (B, 450-520 nm), zeleného (G, 520-600 nm), červeného (R, 630-690 nm) a blízko-infračerveného světla (NIR, 760-900 nm). Rozdíly ve spektrální odrazivosti jednotlivých objektů na scéně jsou poté zdůrazňovány pomocí

vegetačních indexů, definovaných jako matematická kombinace (poměr, lineární vztah) vybraných spektrálních pásem. V dálkovém průzkumu vegetace našli uplatnění zejména metody Normalized Difference of Vegetation Index (NDVI) definovaná jako $NDVI = (NIR - R)/(NIR + R)$ (Rouse, et al., 1973), Ratio Vegetation Index (RVI), daný rovnicí $RVI = NIR/R$ (Jordan, 1969) a Red/Blue Index (Everitt & Villareal, 1987). Při rozpoznání ohnisek plevele a jeho odlišení od pěstované plodiny tyto metody úspěšně použili například Gómez-Casero et al. (2011) či de Castro et al. (2011) s úspěšností rozpoznání vyšší než 90%. Metodu záznamu obrazu v kombinaci blízko-infračerveného světla (NIR, 770–1150nm) a červeného světla (R, 610–670 nm) pomocí bispektrální kamery upevněné na polním robotu také úspěšně využili Weis et al. (2008), přičemž výsledný obraz pro další segmentaci zpracovali pomocí indexu daného rovnicí $R - NIR$.

Hyperspektrální snímkování je potom speciálním případem multispektrálního snímkování, v němž je elektromagnetické záření zaznamenáváno ve velmi úzkých spektrálních pásmech na spojitém spektrálním intervalu, kdy pro každý pixel v obraze je zaznamenáno celé spektrum. Změna odrazivosti v přesně ohraničených spektrálních pásmech byla úspěšně využita zejména pro detekci onemocnění a poškození rostlin (Mahlein, et al., 2011), (Nansen, et al., 2009) či (Polder, et al., 2010) ale i pro segmentaci rostlin a jejich následnou klasifikaci na základě spektrálních charakteristik (Okamoto, et al., 2007). Okamoto et al. popisují možnost rozlišení jednotlivých rostlinných druhů na základě analýzy hyperspektrálního obrazu (Okamoto, et al., 2013).

5.3. Předzpracování obrazu

Metody předzpracování obrazu obecně slouží k usnadnění následného zpracování. Jejich cílem je potlačit šum vzniklý při vytváření obrazu, odstranit zkreslení dané vlastnostmi snímacího zařízení a snímaných objektů, či ke zvýraznění nebo potlačení určitých rysů či objektů.

Mezi nejdůležitější metody předzpracování patří:

a) Převod na stupně šedi.
b) Změna barevného modelu (L*a*b, HSI)

c) Úprava jasu a kontrastu
d) Zaostření obrazu
e) Filtrace

5.3.1. Barevné modely

Většina současných řešení v zemědělství vychází primárně z metod zaznamenání obrazu pomocí technik multispektrálního snímkování. V důsledku toho je předzpracování definováno jako základní kombinace jednotlivých kanálů, shrnuté v předchozí kapitole kvůli realizaci přímo na hardwarovém vybavení. Mimo to byly pro aplikace v zemědělství vyvinuty specifické metody převodu barevného obrazu z běžného fotoaparátu či kamery zaznamenaném ve formátu RGB do stupňů šedi. Často používaná je zejména metoda „*Excessive Green*“ (Woebbecke, et al., 1995) používaná pro odlišení rostlin a půdy (Lamm, et al., 2002), (Meyer, et al., 1998), či (Tang, et al., 2003). Tuto metodu lze zaznamenat pomocí rovnice:

$$(1) \qquad EG = 2 * G - R - B$$

kde *R*, *G* a *B* značí jednotlivé barevné kanály formátu RGB. Tato metoda je také často používána v upravené podobě, kde je každý z barevných kanálů nejprve normalizován pomocí vztahu:

$$(2) \qquad r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}$$

RGB je subtraktivní model barev používaný ve většině současných počítačových řešení pro zobrazování obrazu. Ačkoliv je tento použitelný pro většinu aplikací počítačového vidění, existují i situace, kdy je vhodnější použít některý model, který se více přibližuje intuitivnímu popisu barev tak, jak je intuitivně vnímá člověk, zejména v případě aplikace založené na analýze barev (El-Faki Mozib, et al., 2000). Jako příklady takových modelů mohou být HSI či $L * a * b$.

V barevném modelu HSI nejsou jednotlivé složky popsány pomocí kombinace základních barev, ale pomocí tří základní vlastností barvy – barevným odstínem (hue), sytostí (saturation)

a jasem (intensity). Tento model lze popsat jako válec. Barevný odstín je určen pomocí úhlu od 0° do 360° (0 = červená, 120° = zelená, 240° = modrá), přičemž barvy tvoří uzavřený kruh. Pomocí libovolné výseče je tedy možné omezit požadovaný barevný interval. Sytost je popsaná jako podíl přidané bílé složky. Jas potom vysvětluje, kolik světla barva odrazí, tedy jak zářivá (jasná) bude. Barevný model HSI je definován jako transformace normalizovaného modelu RGB pomocí následujících rovnic:

$$H = cos^{-1}\left(\frac{[(R-G)+(R-B)]}{2[(R-G)^2+(R-B)(G-B)]^{1/2}}\right) \quad (3)$$

$$S = 1 - \frac{3}{I}[\min(R,G,B)] \quad (4)$$

$$I = \frac{1}{3}(R+G+B) \quad (5)$$

Tento model využívá například Tang et al. (2000) pro rozlišení mezi rostlinami a pozadím v podmínkách nestejnoměrného osvětlení na základě barevného odstínu. Metodu HSI využívají také Yiang et al. (Liu, et al., 2007), přičemž dospívají k závěru, že při segmentaci na základě barevné informace je tento model vhodnější než RGB a robustnější vůči změnám v osvětlení, avšak vyžaduje větší množství výpočetních prostředků.

5.3.2. Redukce šumu

Šum je pravidelné (závislé) či náhodné narušení obrazové funkce popisující ideální reprezentace zachycené scény. Při snímání kamerou či fotoaparátem je obvykle přítomen tzv. barevný náhodný šum, způsobený CCD/CMOS snímači použitých zařízení. Tento efekt je obvykle dále znásobený použitou kompresí obrazu (např. formáty .jpeg, .mpeg). Pro zmírnění vlivu této degradace obrazu na následné zpracování jsou využívány rozličné metody odstranění šumu. Při základním zpracování obrazu jsou nejčastěji používané filtry lineární (Gaussovo vyhlazení) a rank-order filtry (mediánový filtr). Při výskytu pravidelného šumu jsou pak výhodné různé varianty odstranění harmonických šumů pomocí vlnkových koeficientů.

Gaussovo vyhlazení je vychází Gaussovy funkce vyjádřené ve dvourozměrném prostoru jako:

$$G(x,y) = \frac{1}{2\pi\sigma^2}\, e^{\frac{x^2+y^2}{2\sigma^2}} \quad (5)$$

Tato funkce je obvykle diskretizována do podoby konvolučního jádra (nejčastěji o rozměrech 3x3, 5x5 či 7x7) a aplikována na obrazovou funkci. Výsledkem Gaussova vyhlazení je rozostřený obraz, který již neobsahuje původní vysokofrekvenční šum kamery. Tato operace však vede k částečném poškození informace v obrazu obsažené, přičemž důsledkem je především snížení ostrosti obrazu na hranách objektů.

Mediánový filtr vyhlazuje obrazovou funkci na základě výpočtu hodnoty mediánu ve stanoveném okolí (obvykle 3x3 či 5x5). Výsledný medián je pak přiřazen centrálnímu bodu okolí. Na rozdíl od filtru Gaussovského zachovává významné hrany a odstraňuje pouze drobné lokální změny v obrazové funkci. Tento filtr je tedy vhodný k odstranění náhodného šumu.

5.3.3. Dekorelace (whitening)

Dekorelace je obecně označení pro metodu využitelnou pro snížení autokorelace signálu. Obrazovou funkci lze považovat za reprezentaci náhodného jevu zachycenou pomocí intenzit na dvourozměrné mřížce. Obrazy zachycují scénu a intenzity sousedících bodů jsou mezi sebou silně korelované, což může způsobit ztrátu (či zanedbání) důležité informace v obrazu obsažené. Dekorelace obrazové funkce je obvykle realizována na malých výřezech vstupního obrazu (12x12, 16x16) (Ng, et al., 2015) , přičemž snahou je minimalizovat korelaci mezi jednotlivými složkami vzniklých vektorů intenzit. Tyto výřezy jsou vybrány náhodně z učební sady, důležité je, aby vybrané vzorky byly reprezentativním výběrem rozličných výřezů.

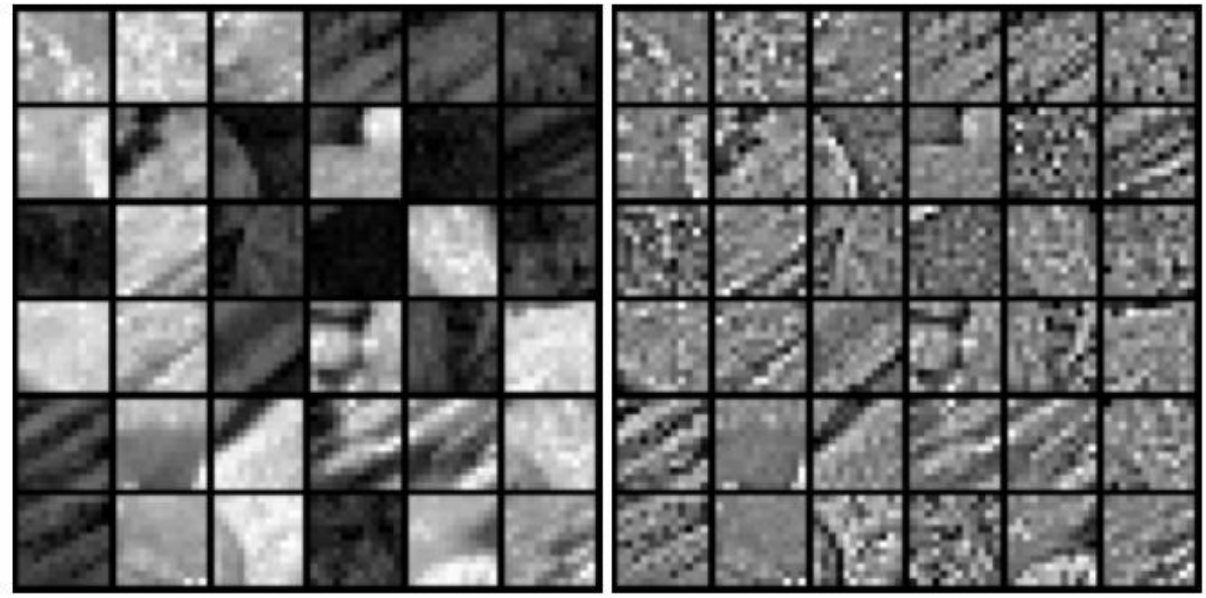
Dekorelace obrazové funkce je transformací obrazové funkce tak, že její kovariační matice $\sum$ je maticí identity. Je-li dáno N bodů v R^n, potom jejich kovariační matice $\sum \in R^n$ je vypočtena jako:

$$\hat{\Sigma}_{jk} = \frac{1}{N-1}\sum_{j=1}^{N}\left(x_{ij} - \bar{x}_j\right)\cdot\left(x_{ik} - \bar{x}_k\right), \quad (6)$$

kde $\bar{x}_j$ označuje j komponent průměru vzorku *x*. Pro dekorelaci takového výběru lze použít libovolnou matice $W \in R^n$ která splňuje rovnici:

$$w^T w = C^{-1}. \quad (7)$$

Toho lze dosáhnout výpočtem vlastních vektorů kovariační matice a popisem použitých výřezů pomocí soustavy rovnic, která je určena vlastními těmito vektory. Pro dekorelaci jsou v současnosti nejpoužívanější techniky založené na analýze hlavní komponenty (PCA), dekompozici Choleskyho matice a analýze nulové fáze (ZCA) (Kessy, et al., 2015).



Obrázek 1: Ukázka dekorelace za přítomnosti šumu (volně převzato z (Ng, et al., 2015))

Dekorelace vede k zdůraznění vlivu šumu, což snižuje přesnost a použitelnost následujících algoritmů segmentace a extrakce příznaků. Je proto vhodné nejprve obrázek od šumů očistit, pomocí některé z technik uvedených v předcházející kapitole.

5.4. Segmentace

Segmentace obrazu je definována jako rozdělení obrazu na množinu nepřekrývajících se regionů, jejichž sjednocením vznikne původní obraz. Cílem segmentace je rozklad obrazu na části, které jsou smysluplné s ohledem na konkrétní aplikaci (Haralick & Shapiro, 1992). Cílem je tedy obvykle zjednodušení původního obrazu na reprezentaci, které je vhodná pro další obrazovou analýzu (Barghout & Lee, 2003). Segmentace může být popsána jako procedura, která seskupuje části obrazu na základě společných charakteristik, jako jsou barva (intensita, odstín, jas) či sémantický vztah k dalším objektům v obrazu (na základě textury, lokálních

hranic, či okolí). Výsledné sousední regiony se významně liší vzhledem k některé z těchto charakteristik (Haralick & Shapiro, 1992).

Segmentace obrazu daného funkcí $f(x, y)$ lze matematicky definovat jako rozložení obrazu na *n* regionů *R* takovým způsobem, že tyto regiony splňují následující podmínky:

a) $\bigcup_{i=1}^{n} R_i = f(x, y)$
b) $R_i \cap R_j = \emptyset, i \neq j$
c) Každý pixel v regionu R_i splňuje stanovené kritérium (náleží do intervalu intenzity, jeho intenzita je vyšší než stanovený práh, nachází se mezi hranami, aj.).

Segmentace je nejčastěji používána pro rozlišení objektů a jejich pozadí a označení jejich přirozených ohraničení (kontur, hran). Segmentačních metod a algoritmů dnes existují tisíce (Zhang, 2006), obvykle založené na principu studia vlastností jednotlivých pixelů, detekci hran, vlastností regionů či kombinací těchto přístupů (Dey, et al., 2010). Tento přehled zmiňuje pouze algoritmy, které jsou běžnou součástí řešení používaných pro rozpoznání plevelů.

5.4.1. Prahování

Prahování (angl. *thresholding*) je základní segmentační metoda založena na předpokladu, že je možné oddělit popředí a pozadí na základě odlišné intenzity. Je tedy možné najít takovou prahovou intenzitu, kterou je možné použít pro rozdělení obrazu do několika částí. V nejzákladnější podobě je intenzitní obraz f převeden na binární obraz g (Shapiro & Stockman, 2002), dle vztahu:

$$g(i,j) = \begin{cases} 1 \; pro \; f(i,j) \geq \theta \\ 0 \; pro \; f(i,j) < \theta \end{cases}, \quad (8)$$

kde θ je práh. Práh může být použit buď globálně (jedna prahová hodnota pro celý obraz) nebo lokálně, kdy je práh určen pro jednotlivé části obrazu. Obě možnosti mají své nedostatky. Globální prahování je citlivé na nerovnoměrné osvětlení, zatímco adaptivní práh často vede k neočekávaným výsledkům, a vytváří nespojitosti v sousedících regionech.

V závislosti na segmentační úloze je také možné zachovat intenzitu popředí (popř. pozadí) pomocí operace poloprahování:

$$(9) \qquad g(i,j) = \begin{cases} f(i,j)\ pro\ f(i,j) \geq \theta \\ 0\ pro\ f(i,j) < \theta \end{cases}$$

Je také možné rozdělit obraz do většího množství nalezení několika mezních prahových hodnot, na jejichž základě je obraz segmentován. Tento postup lze aplikovat jak na globální úrovni pro celý obraz, tak na úrovni lokální pro jeho jednotlivé sub-obrazy. Ve všech zmíněných případech je výsledek závislý na správné volbě mezní prahové hodnoty, která složí k rozlišení jednotlivých regionů v obrazu. Zevrubný přehled metod pro volbu prahové hodnot vypracovali Sezgin a Sankur (2004), následující přehled uvede pouze vybrané algoritmy často používané pro segmentaci obrazu v zemědělských aplikacích.

Mnoho algoritmů využívá pro nalezení optimální prahové hodnoty analýzy histogramu intenzitního obrazu. Základní hypotézou potom je předpoklad, že je-li v obrazu rozlišitelné popředí a pozadí, odráží se tento fakt i v histogramu, který má charakter bimodální (popř. multimodální) křivky. Prahová hodnota se potom nejčastěji nachází v údolí mezi dvěma vrcholy histogramu. Příkladem takového algoritmu je například metoda založená na iterativním vyhlazování histogramu (Prewitt & Mendelsohn, 1966), přičemž vyhlazování je prováděno tak dlouho, dokud nevznikne jedno lokální minimum mezi dvěma vrcholy. Tento bod je následně vybrán jako práh. Analýzy histogramu využívají také metody založené na překryvu konkávních částí histogramu pomocí konvexního obalu (Sezgin & Sankur, 2004).

Dalším typem algoritmů jsou potom takové, které analyzují intenzitu obrazu pomocí statistických technik. Příkladem takového algoritmu je *Otsuova metoda maximálního rozptylu* (Otsu, 1979), která stanovuje prahovou hodnotu pomocí celkové střední hodnoty a rozptylu pole intenzit obrazu. Otsu definoval odchylku mezi dvěma třídami segmentovaného obrazu pomocí diskriminační analýzy jako:

$$(10) \qquad {\sigma_B}^2 = w_1(\mu_1 - \mu_\tau) + w_2(\mu_2 - \mu_\tau)\ ,$$

kde μ_1 a μ_2 jsou průměry dvou regionů oddělených prahovou hodnotou. Optimální práh je následně vybrán na základě maximalizačního kritéria:

(11) $t^* = ArgMax\,\{\sigma_B^2(t)\}$

Dalšími možnostmi jsou pak algoritmy využívající entropie cross-entropie mezi původním a binárním obrazem nebo metody využívající vyššího řádu rozdělení pravděpodobnosti nebo korelace mezi pixely ve stanoveném okolí (Sezgin & Sankur, 2004).

Metody prahování jsou obecně použitelné v situacích, kdy je snadné rozlišit mezi popředím a pozadím, či na lokální bázi.

5.4.2. Detekce hran

Metody detekce hran jsou obecně založeny na předpokladu, že na hranicích regionů dochází ke změně vlastností obrazu. Místa v obraze, která odpovídají významným hranám tak nesou více informace než regiony homogenního charakteru. Hrany v obraze vznikají především díky nespojitostem v povrchu, jeho odrazivosti, odleskům nebo výskytu stínů. Hrana obecně popisuje směru největšího růstu funkce obrazu $f(x, y)$, tedy její gradient. Tohoto faktu také využívá většina hranových detektorů. Tyto lze obecně rozdělit:

- Detektory založené na hledání maxim prvních derivací
- Detektory založené na hledání průchodu druhých derivací nulou
- Lokální aproximace funkce obrazu parametrickým modelem

Detektory založené na hledání maxim prvních derivací jsou založeny na měření gradientu obrazové funkce *f* v určitém umístění. Toto je ve strojovém vidění, realizováno prostřednictvím gradientních operátorů. Tyto operátory, také označované jako masky jsou aproximací ortogonálních gradientních vektorů f_x a f_y. Nechť je *H* maska o velikosti *p x p*, v níž bod *x (m, n) označuje* libovolnou pozici v obrazu. Každý gradientní operátor je potom reprezentován dvojicí masek H_1 a H_2, které měří gradient v obrazu v bodě *x* ve dvou ortogonálních směrech. Mezi nejpoužívanější operátory potom patří následující operátory:

Sobelův operátor: $H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ $H_2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

Operátor Prewittové: $H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ $H_2 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

Je-li výchozí obraz definovaný jako funkce $f(m,n)$, G_1 a G_2 jsou aproximací derivace v horizontálním a vertikálním směru spočtené jako:

(12) $G_1(m,n) = f(m,n) * H_1(m,n)$

(13) $G_2(m,n) = f(m,n) * H_2(m,n)$

kde * značí dvourozměrnou konvulzní operaci. Pro každý bod v obrazu lze pak spočítat výslednou aproximaci gradient jako kombinace horizontální a vertikální složky jako:

(14) $G(m,n) = \sqrt{G_1^2(m,n) + G_2^2(m,n)}$

Sobelův algoritmus, algoritmus Prewittové i další konvulzní jádra založená na metodě první derivace zaznamenávají veškeré hrany v obrazu, včetně hran nevýznamných. Jako rozšíření těchto základní přístupů vznikl algoritmus navržený Johnem Cannym (Canny, 1986), který spojuje algoritmus detekce hran s eliminací šumu, a následnou eliminaci nevýznamných hran. Skládá se ze 4 základních kroků:

- Eliminace šumu pomocí Gaussova filtru
- Výpočet směru a intenzity gradientu (obvykle pomocí Sobelova operátoru)
- Nalezení pixelů, které skutečně tvoří hranu, na základě analýzy lokálních maxim v okolí (thinning)
- Eliminace nevýznamných hran, na základě analýzy, kde hrany začínají a končí

Další skupinou algoritmů pro detekci hran jsou algoritmy založené na vyšších derivacích, které obecně vycházejí z Laplaciánů obrazové funkce, respektive z operátoru označovaného jako LoG (Laplacian of Gaussians). Typickým příkladem takového algoritmu je potom hranový detektor Marr-Hildrethové.

5.4.3. Metody využívající algoritmů strojového učení

Většina výše uvedených metod segmentace je založena pouze na studiu hodnot intenzity pixelů v obrazu, popřípadě studiu vlastností okolí, které je dané velikostí konvulzního jádra. Přirozená podstata segmentační úlohy je ale obvykle komplikovanější. Jako možná řešení byly navrženy rozličné metody umělé inteligence. Příkladem je například metoda GAHSI (Tang, et al., 2000), která kombinuje barevný prostor HSI a genetický algoritmus pro rozlišení rostliny a půdy. Shrestha et al. (2001) potom používají neuronovou síť pro odhad tvaru elipsoidu pro segmentaci barevného obrazu na základě středních hodnot a směrodatných odchylek pro jednotlivé kanály v RGB jako vstupu a ručně zvolených parametrů jako cílového výstupu, přičemž na testovací množině dosahují výsledků srovnatelných s ruční segmentací.

Segmentaci je také možné provádět pomocí metod klastrování, popřípadě pomocí kompetičních sítí. Speciální pozornost si pak zaslouží metody využívají konvoluční neuronové sítě.

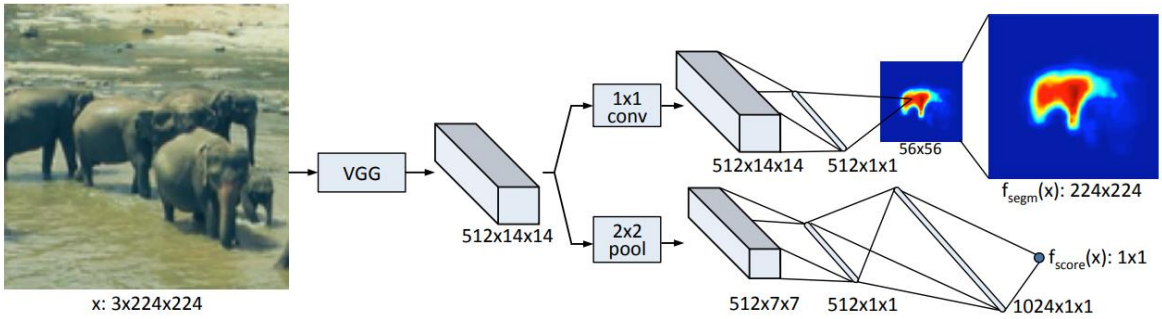
5.4.4. Konvoluční neuronové

Segmentace objektů pomocí konvolučních neuronových sítí lze považovat za rozšíření problému lokalizace objektu. Cílem je pro každý z obrazů v obrazu přiřadit odpovídající bitmapovou masku. Konvoluční sítě a jejich obecné principy vysvětluje ve více detailech kapitola 5.7.

Fully Convolutional Networks (FCN) neboli plně konvoluční sítě nahrazují běžně používané závěrečné plně propojené vrstvy sítě vrstvami konvolučními. (Long, et al., 2015). Výstupem takové sítě je dvourozměrná mapa vektorů s rozlišením odpovídajícím poslední běžné konvoluční vrstvě. Tento výstup lze použít jako hrubou segmentační masku. Při zmenšení posunu (stride) v poolingových vrstvách lze dosáhnou poměrně vysoké kvality segmentace (Long, et al., 2015). Tato architektura je v současném stavu schopna segmentovat (a tedy rozlišit) třídy objektů.

Architektura DeepMask (Pinheiro, et al., 2015) vychází ze sítě VGG, ze které odstraňuje plně propojené vrstvy. Vrstvy sloužící pro extrakci příznaků jsou tedy zachovány a po nich je síť rozdělena na dvě větve – první z nich predikuje segmentační masku pro objektu umístěný uprostřed scény, zatímco druhá část vyhodnocuje skóre pro přítomnost či částečnou přítomnost

objektu ve scéně. Pro učení sítě je využívána trojice vstupů: originální obraz, binární masku, která odpovídá objektu v obrazu obsaženém a označení, zda je objekt přítomen ve středu obrazu či nikoliv. Tato architektura je schopna nalézt v obrazu vícero výskytů stejné třídy objektu (např. několik osob, zvířat), přičemž každý výskyt bude mít na snímku vlastní masku.



Obrázek 2: Ukázka architektury DeepMask (Pinheiro, et al., 2015)

Architektura SharpMask (Pinheiro, et al., 2016) je rozšířenou a vylepšenou variantou sítě DeepMask, jejímž cílem je dosáhnout vyšší přesnosti segmentačních masek. Toho je dosaženo implementací zpětného propojení na vyšší konvoluční vrstvy výchozí VGG architektury, které mají vyšší rozlišení. Jedná se tak v podstatě o operaci, která je schopna odstranit vliv downsamplingu v poolingových vrstvách na konečný výsledek, zdůraznit jednotlivé hrany a přechody a dosáhnout tak výrazně vyšší přesnosti segmentační masky.

5.5. Extrakce příznaků, popis objektů v obraze

Uvažujme nyní binární obraz f, který je výstupem segmentace obrazu. Při rozpoznání a klasifikaci objektů v obrazu obvykle narážíme na problém degradace obrázku díky neideálním podmínkám snímání obrazu. Takový obraz g lze tedy popsat pomocí funkce:

$$g = D(f) \tag{15}$$

Kde D je neznámý operátor, popisující degradaci obrazu. Typickými degradacemi jsou potom otočení a změna měřítka obrazu, afinní transformace obrazu, neúplnost obrazu, či degradace, které lze popsat jako výsledek konvoluce. Pro praktické aplikace je tedy nutné najít řešení, které je schopné klasifikace i za podmínek degradace. Jako možná se nabízí:

- Řešení hrubou silou
- Popis obrazu pomocí invariantů
- Nalezení normalizované pozice obrazu (úloha inverzní k popisu pomocí invariantů)

Řešení hrubou silou vyžaduje, aby pro klasifikaci byla použita množina, která obsahuje příklady co největšího možného množství obrazů, včetně obrazů obsahujících degradace. Takový postup vyžaduje velkou učební množinu, zejména v podmínkách přírodních scén, a je tedy značně komplikovaný i za použití idealizovaných scénářů, a prakticky nepoužitelný pro úlohy strojového učení. Jako možné řešení je tak často využíván popis objektů pomocí příznaků či invariantů.

Invariant I je funkcionál definovaný tak, že platí $I(f) = I(D(f))$ pro veškerá přípustná D. Zároveň pak platí, že hodnoty invariantů jsou odlišné pro rozdílné kategorie objektů. Invarianty lze obecně rozlišit:

- Jednoduché tvarové deskriptory
- Kompaktnost, konvexnost, elongace …
- Transformační koeficienty
- Fourierovy deskriptory, vlnkové vzory a funkce
- Bodové množiny
- Pozice dominantních bodů
- Diferenciální invarianty
- Derivace hrany
- Momentové invarianty

Následující přehled se bude zabývat zejména takovými typy invariantů, které byly úspěšně použity při klasifikaci rostlin.

5.5.1. Jednoduché tvarové deskriptory

Jednoduché tvarové deskriptory někdy také označované jako *hand-crafted features* (např. (Hall, et al., 2015)) jsou číselné charakteristiky objektů vycházející z přirozeného vnímání tvaru a jeho vlastností, jako je plocha, obvod, výška, šířka či členitost. Invariantnost těchto hodnot je

potom zajištěna pomocí poměrové kombinace jednotlivých vlastností, která zajišťuje jejich nezávislost na rotaci a pozici. Některé z těchto vlastností lze snadno spočítat algoritmicky, včetně plochy objektu (a) a jeho obvodu (p), velikosti konvexního obalu (c), a obvodu konvexního obalu (r), délky kostry (s), či největší vzdálenosti mezi libovolnými dvěma pixely na hraně objektu (d). Některé však spočívají v manuálním popisu objektu, nebo jeho umístění v předem známé pozici, včetně vlastností jako je největší přirozená výška (h) a šířka (w).

Tabulka 1 uvádí některé běžně používané tvarové deskriptory, které (často s množinou dalších příznaků) s rozličnou úspěšností využívají při klasifikaci listů Kadir et al. (2011) (5-7), Hall et al., (2015) (1-4) či Chaki (2015) (5,6,8) či Wu et al. (Wu, et al., 2007) (1-3, 5, 8-10).

	Deskriptor	**Výpočet**	**Popis.**
1	Kompaktnost	a/p^2	Poměr plochy objektu a kvadrátu jeho obvodu
2	Jednolitost	a/c	Poměr plochy objektu a jeho konvexního obalu
3	Konvexnost	p/r	Poměr obvodu objektu a obvodu jeho konvexního obalu
4	-	s/p	Poměr délky kostry objektu a jeho obvodu
5	Kulatost	$\frac{4\pi a}{p^2}$	Podobnost mezi tvarem listu a kruhem
6	Poměr stran	h/w	Poměr přirozené délky a šířky tvaru
7	Nepravidelnost (Nixon & Aguado, 2002)	$\frac{\max(\sqrt{(\mathrm{x_i}-\bar{\mathrm{x}})^2+(\mathrm{y_i}-\bar{\mathrm{y}})^2})}{\min(\sqrt{(\mathrm{x_i}-\bar{\mathrm{x}})^2+(\mathrm{y_i}-\bar{\mathrm{y}})^2})}$	Poměr největší a nejmenší vzdálenosti od bodu na hraně (x_i, y_i) k těžišti $(\bar{x}, \bar{y})$
8	Hranatost	$\frac{h*w}{a}$	Podobnost mezi tvarem a obdélníkem daným přirozenou výškou a šířkou tvaru

9	-	d/h	Poměr mezi maximální vzdáleností mezi dvěma body na hraně objektu a jeho přirozenou délkou
10	-	p/d	Poměr obvodu a maximální vzdáleností mezi dvěma body na hraně objektu

Tabulka 1: Jednoduché tvarové deskriptory

Tvarové deskriptory tohoto typu jsou oblíbené zejména pro jejich snadnou a rychlou implementaci. Použitelnost takovéhoto druhu příznaku je potom dána zejména charakterem úlohy. Jsou-li rozdíly mezi klasifikovanými objekty významné, mohou být tyto deskriptory vhodnou metodou popisu objektu.

5.5.2. Momenty a momentové invarianty

Momentové invarianty jsou funkce momentů obrazu, které jsou neměnné vzhledem k určitému typu degradace. Momentové invarianty jsou definované s pomocí typu degradace obrazu, vůči němuž jsou neměnné. Nejběžnější skupiny pak zahrnují:

- Rotace, překlady, škálování
- Afinní transformace
- Elastické deformace
- Konvulze/rozostření
- Kombinované invarianty

Momenty obrazu jsou potom definovány jako projekce funkce obrazu na polynomiální bázi. Mějme funkci f (x, y) a množinu polynomů $P_{pq}(x, y)$ definovaných na $\Omega \subset$ R x R. Moment lze pak obecně vyjádřit pomocí následující rovnice:

(16) $$m_{pq} = \iint P_{pq}(x, y) f(x, y) dx dy$$

Nejčastěji používanými typy momentů jsou potom momenty geometrické, definované v diskrétní podobě jako:

(17) $M_{pq}^{(f)} = \sum_x \sum_y x^p y^q f(x,y)$

Předpokládáme-li, že řád momentu je daný součtem p a q, lze pomocí momentu nultého řádu popsat plochu objektu, za použití metod prvního řádu lze spočíst těžiště objektu, momenty druhého řádu složí k popisu momentů setrvačnosti, a momenty třetího řádu pak slouží k popisu špičatosti objektu. Praktického využití se pak dočkávají zejména momenty v normalizované podobě, zejména pak momenty centrální dané rovnicí:

(18) $\mu_{pq}^{(f)} = \sum_x \sum_y (x-\bar{x})^p \, (y-\bar{y})^q f(x,y) \quad kde \;\; \bar{x} = \frac{M_{10}}{M_{00}} \;\; a \;\; \bar{y} = \frac{M_{01}}{M_{00}}$

Bod $(\bar{x}, \bar{y})$ je potom tedy těžištěm popsaného objektu. Centrální momenty jsou invariantní vůči rotaci a změně pozice (translaci). Pro dosažení nezávislosti na velikosti objektu je potom nutné centrální momenty normalizovat pomocí vzorce:

(19) $\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(1+\frac{i+j}{2}\right)}}$, kde $i+j > 2$

Huovy invariantní momenty (Hu, 1962) jsou množina deskriptorů objektu, které jsou invariantní vůči translaci a v normalizované podobě také vůči změnám ve škále a rotacím. Tyto momenty jsou vypočítány z normalizovaných centrálních momentů do třetího řádu jako:

(20) $I_1 = \eta_{20} + \eta_{02}$

(21) $I_2 = (\eta_{20} - \eta_{02})^2 + \; 4\eta_{11}^2$

(22) $I_3 = (\eta_{30} - 3\eta_{12})^2 + \; (3\eta_{21} - \eta_{03})^2$

(23) $I_4 = (\eta_{30} + 3\eta_{12})^2 + \; (\eta_{21} + \eta_{03})^2$

(24) $I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + 3\eta_{12})^2 - \; 3(\eta_{21} + \eta_{03})^2] \; + (3\eta_{21} - \eta_{03}) \; (\eta_{21} + \eta_{03})[3(\eta_{30} - \eta_{12})^2 - \; (\eta_{21} - \eta_{03})^2]$

(25) $I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} - \eta_{12})(\eta_{21} - \eta_{03})]$

(26) $I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] + (\eta_{30} + 3\eta_{12})(\eta_{21} - \eta_{03})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2]$

Huovy momenty jsou často používaným příznakem při rozpoznání rostlin a objektů obecně na základě jejich tvarových vlastností. Kadir et al. (2011) dosáhli úspěšnosti klasifikace listů stromů až 72,40 %.

5.5.3. Fourierovy deskriptory

V mnoha případech je vhodné popsat objekt v obraze tak, že výchozím bodem pro jeho popis je jeho těžiště. Takové případy jsou obecně označovány jako transformace obrazu z dvojrozměrného Kartézského prostoru do polárního prostoru. Je-li obraz daný jako $I = \{f(x,y);\ 0 \le x \le M, 0 \le y \le N\}$ v Kartézském prostoru, v polárním prostoru bude zaznamenán jako funkce vzdáleností hrany objektu od těžiště pod daným úhlem θ jako$I_p = f(r,\theta);\ 0 \le r \le R, 0 \le \theta \le 2\pi$, kde R je maximální vzdálenost mezi těžištěm a hranou objektu.

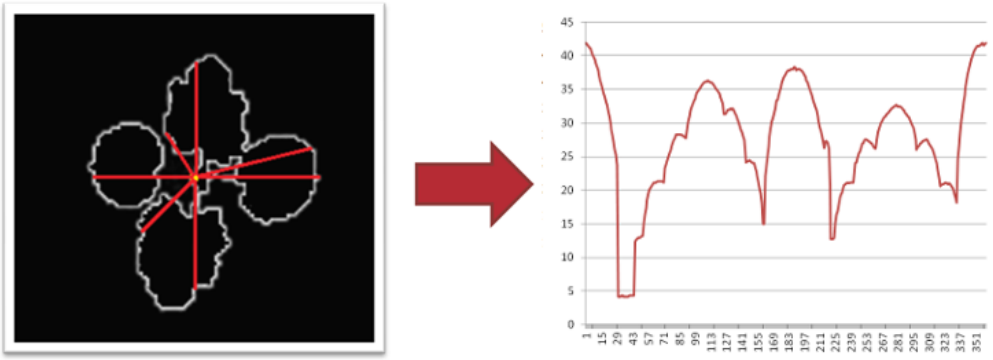
Obecná Fourierova transformace do polárních souřadnic je potom diskrétně vyjádřena jako: $PF(\rho,\Phi) = \sum_r \sum_i f(r,\theta_i)\ \exp[-j2\pi\ (\frac{r}{R}\rho + \frac{2\pi i}{T}\Phi)]$, kde $0 \le r < R$ a $\theta i = i\left(\frac{2\pi}{T}\right) (0 \le i < T); 0 \le \rho < R, 0 \le \phi < T$. R je potom radiální frekvenční rozlišení a T maximální úhlová frekvence. Takto vyjádřený deskriptor je invariantní vůči translaci. Pro invarianci k velikosti je ještě nutné normalizaci, čímž vznikne Obecný Fourierův deskriptor (GFD) vyjádřený pomocí rovnice:

(27) $GFD = \left\{\frac{|\mathrm{PF}(0,0)|}{\mathrm{M}_{00}}, \frac{|\mathrm{PF}(0,1)|}{|\mathrm{PF}(0,0)|}, \dots, \frac{|\mathrm{PF}(0,n)|}{|\mathrm{PF}(0,0)|}, \dots, \frac{|\mathrm{PF}(m,n)|}{|\mathrm{PF}(0,0)|}\right\}$,

Kde M_{00} je geometrický moment udávající plochu objektu. Podobnost mezi dvěma objekty vyjádřenými pomocí GFD je potom obvykle měřena jednoduchými mírami, jako je Hammingova vzdálenost. Kadir et al. (2011) pomocí tohoto postupu dosahují přesnosti klasifikace listů až 93%.

5.5.4. CCD / Centroid-Radii

Transformace do systému polárních souřadnic využívají také další populární tvarové deskriptory jako Contour centre distance (CCD) nebo Centroid-Radii (Tan, et al., 2003), které spočívají v nalezení průsečíku polopřímky vycházející z těžiště objektu s jeho hranou. Výsledná vzdálenost je následně změřena a celé měření opakováno pod stanoveným počtem úhlů. Tato transformace je schematicky naznačena v obrázku 1.



Obrázek 3: Transformace do systému polárních souřadnic měřena jako vzdálenost pixelu na hraně od těžiště

Formálně pak je možné zapsat tento deskriptor popsat jako n-prvkovou množinu:

$$S = \{r_0, r_\theta, r_{2\theta}, \dots r_{2\pi-\theta}\} \quad (28)$$

kde θ je úhel rotace z intervalu $< 0{,}2\pi >$ a r je Euklidovská vzdálenost mezi těžištěm objektu a bodem na hraně. Tato metoda byla úspěšně použita při rozpoznání listů, jejichž plocha je celistvá (Chaki & Parekh, 2011). Při použití objektů, jejichž plocha je výrazně členitá však selhává, díky významné ztrátě informace (Chaki, et al., 2015). ¨

5.5.5. Scale Invariant Feature Transform (SIFT)

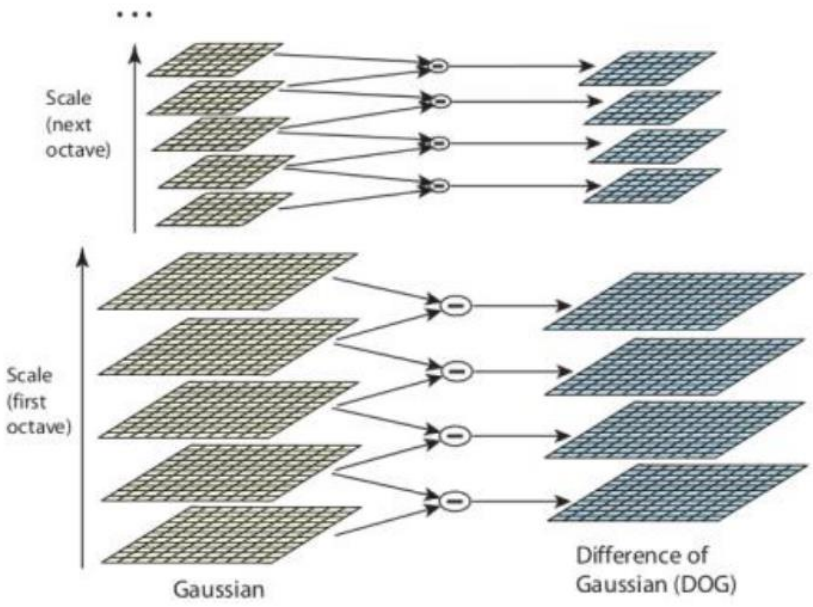
SIFT (Lowe, 1999) je algoritmus extrakce příznaků, který z obrazu vybírá množinu význačných bodů a na základě těchto bodů vypočte specifické deskriptory. Tyto deskriptory lze odlišit od ostatních bodů i v případě, že je objekt (scéna) snímán z jiného pohledu. Získané deskriptory

obrazu jsou invariantní vůči změnám velikosti, rotace a jasu, částečně pak vůči dalším afinním transformacím.

Algoritmus je obecně možné rozdělit od pěti kroků:

- Vytvoření scale-space pyramidy
- Aproximace LoG pomocí DoG
- Detekce klíčových bodů a eliminace nevhodných bodů
- Přiřazení orientace klíčovým bodům
- Extrakce deskriptoru

Prvním krokem je vytvoření tzv. scale-space pyramidy, která je základem invariance algoritmu vůči změnám velikosti. Pro potřeby této pyramidy je obrázek postupně zmenšován (resp. Klesá jeho rozlišení), kdy následující vrstva je vždy velikostně polovinou vrstvy předchozí. Tyto vrstvy jsou také označovány jako oktávy. V rámci každou oktávu je opakovaně aplikováno Gaussovo rozostření se zvyšující se hodnotou parametru σ (viz. kapitola 5.3.2), díky čemuž je obraz postupně rozostřován.



Obrázek 4: Konstrukce space-scale pyramidy a převod na diference Gaussiánů [převzato z (OpenCV, 2015)]

V druhé fázi jsou od sebe vzájemně odečteny sousední snímky v oktávě, čímž je vypočten rozdíl dvou Gaussiánů (DoG). Tento rozdíl lze považovat za aproximaci filtru Laplacián Gaussiánu, který je výpočetně významně náročnější. Ve výsledných snímcích diferencí jsou následně nalezena lokální maxima, a pro každé z těchto maxim je vypočtena orientace na základě histogramu gradientů.

Závěrečným krokem je získání deskriptoru významných bodů založené na popisu okolí detekovaných lokálních maxim. Je vybráno okolí o rozměrech 16x16 kolem klíčového bodu, které je dále rozdělená na 16 menší výřezů o velikost 4x4. Pro každý takto vzniklý výřez vzniká histogram gradientu o osmi třídách, pro popis každého bodu je tedy vytvořeno 128 popisných bodů (4x4x8). Takto vzniklé histogramy jsou také váženy, přičemž vyšší váha je přiřazena těm, které jsou blíže významnému bodu. Tato úprava zajišťuje snižuje citlivost vůči degradacím obrazu (Rey Otero & Delbracio, 2014).

5.5.6. Speed-up Robust Features (SURF)

SURF (Bay, et al., 2006) je algoritmus extrakce příznaků částečně inspirovaný algoritmem SIFT. Tento algoritmus je obdobně robustní vůči rotaci, posunům a změně velikosti objektů, jakož i vůči dalším afinním transformacím (např. změna úhlu z něhož je scéna zabíraná). Hlavní motivací pro vývoj tohoto algoritmu byla rychlost výpočtu, důležitá zejména pro online výpočty a zpracování video sekvencí.

SURF využívá kvůli rychlosti výpočtů převodu standardní reprezentace na obraz integrální. Dalším zrychlením je pak nahrazení Gaussova filtru pro rozostření aproximací, která tento filtr aproximuje, tzv. box filtrem při konstrukci space-scale pyramidy. Rozdílem je pak také to, že na rozdíl od algoritmu SIFT, který vždy zmenšuje velikost následující vrstvy, SURF využívá rostoucí filtr. Detekce významných bodů je pak založena na výpočtu determinantu Hessovy matice, která je použita jako míra změn v okolí bodu. Jako významné body jsou zvoleny ty, kde determinant nabývá maxima.

Orientace gradientu takto nalezeného klíčového bodu je určena sumou odezev Haarových vlnek. Sledování orientace gradientů zajišťuje invarianci vůči rotacím. Při výpočtu deskriptoru je oblast kolem klíčového bodu rozdělena na podoblasti o velikosti 4x4 pixelů, přičemž pro

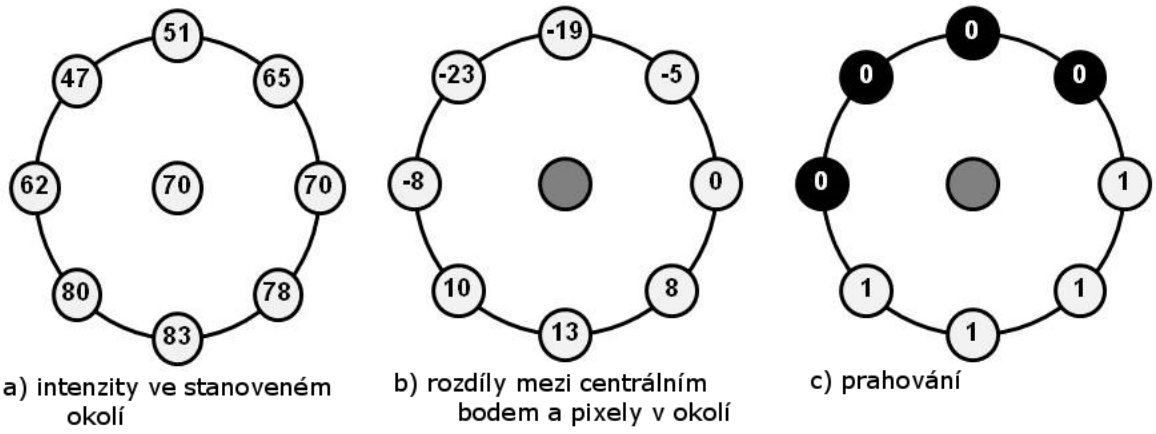
každou z těchto oblastí je vypočtena suma odezev Haarovy vlnky ve vertikálním a horizontálním směru a jejich suma absolutních hodnot. Výsledkem je 64 popisných bodů, tedy polovina velikosti deskriptoru v algoritmu SIFT.

5.5.7. Local Binary Patterns (LBP)

Local Binary Pattern je deskriptor obrazové funkce, který zaznamenává strukturu okolí zvoleného bodu v obraze pomocí porovnání intenzit a jejich následnou transformací do podoby binárního řetězce dle následující rovnice:

$$(29) \quad LBP_{P,R}(x,y) = \sum_{i=0}^{P-1} 2^i \,\mathrm{thr}\,\big(I(x_i,y_i) - I(x,y)\big), \quad thr(x) = \begin{cases} 1, & je-li\ x \geq 0; \\ 0, & je-li\ x < 0. \end{cases},$$

Kde P udává délku binárního řetězce, R poloměr použitého okolí kruhového tvaru (Ojala, et al., 2002), nejčastěji o poloměrech 3, 5, 8 či 16 bodů od středového pixelu. Bod na souřadnicích [x, y] je pak centrálním bodem toho okolí, každý z bodů se souřadnicemi $[x_i, y_i]$ pak leží na kružnici dané poloměrem R a centrálním bodem. Výsledná řada binárních hodnot je pak interpretována jako celé číslo, které reprezentuje vlastnosti intenzity v okolí zvoleného bodu.



Obrázek 5: Ukázka postupu výpočtu bitového řetězce P = 8 pomocí kruhového okolí

Pro každý bod v obraze je tedy možné stanovit popis intenzity v okolí bodu. Na základě toho je také možné spočíst histogram popisující celý oblast, či jen vybraný výřez. Výsledkem je při použití 8 bodů pro charakterizaci textury histogram obsahující 256 kategorií.

Ojala et al. (1994) uvádí, že naprostá většina textur, které lze v obrazech popsat má jednu společnou vlastnost: v kruhovém binárním okolí (viz. obrázek 5.c výše) obsahují nejvýše 2 přechody z 0 na 1. Takové vzory označujeme jako jednotné (uniformní). Jelikož je možné většinu informací vysvětlit pomocí uniformních vzorů, je možné histogram zkrátit a využít pouze n 58 odlišných uniformních zdrojů a jeden zásobník pro veškeré vzory neuniformní.

Při použití LBP operátoru s větším poloměrem může vlivem aliasingu a šumu docházet k tomu, že výsledná hodnota nebude adekvátní reprezentací textury. Proto je často na obraz aplikován nízko frekvenční Gaussovský filtr, který vliv těchto negativních jevů sníží (Mäenpää, 2003).

Metoda LBP byla úspěšně využita zejména pro segmentaci textur, ale lze ji využít i pro jejich rozpoznání.

5.5.8. Automatická extrakce příznaků, sparse coding a autoencoding

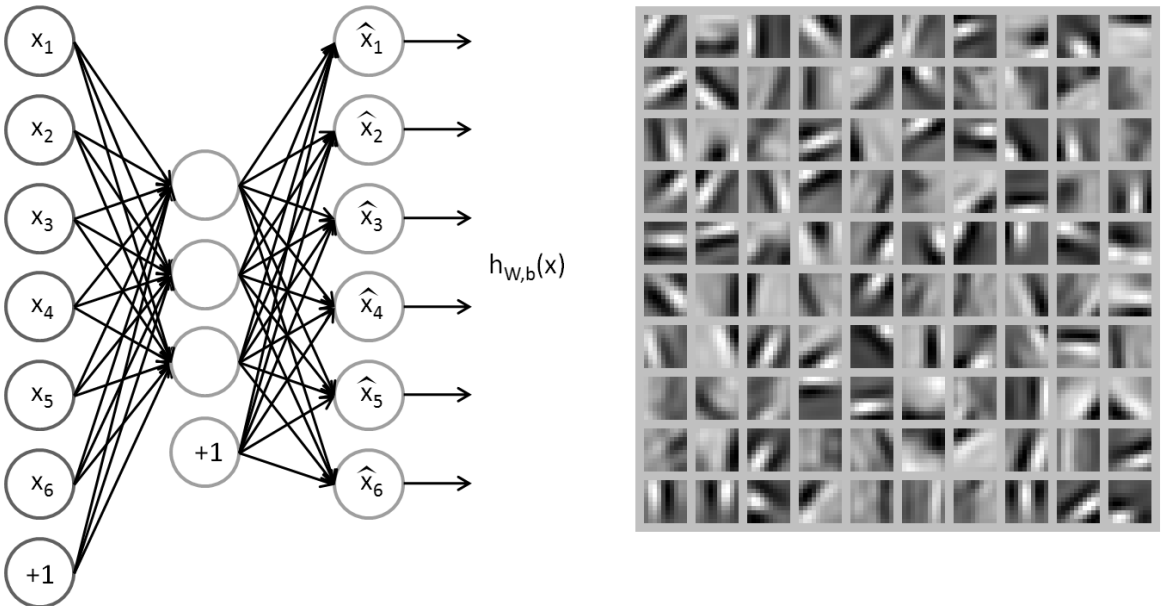
Algoritmy uvedené v jednotlivých podkapitolách kapitoly 5.5. jsou založeny na expertním popisu a následné extrakci vlastností, které jsou vhodné pro rozpoznání obrazu. Důraz je pak kladen zejména na invarianci vůči nejběžnějším typům deformací obrazu. Většina těchto metod je však náchylná k poměrně významné ztrátě informace, která je pro rozpoznání objektu důležitá.

Metody automatické extrakce naproti tomu vybírají přímo autonomně, přímo na základu učební sady. Pro extrakci je obraz rozdělen do menších rámců o velikosti 10x10 pixelů (někdy až 20x20). Tyto výřezy je možné považovat za bloky, z nichž se každý obraz skládá. Cílem extrakce je nalézt takovou bázi pro tyto vektory, aby se co nejvíce koeficientů jejich lineární kombinace blížilo nule. Výsledná báze se označuje jako tzv. řídké kódování (angl. *sparse coding*) a obsahuje vektory, které odpovídá vzorům, které se v obraze často vyskytují, jako jsou orientované hrany či rohy (Olshausen & Field, 1997). Řídké kódování lze vyjádřit pomocí rovnice jako:

(28) $$\vec{x} \approx \sum_{i=1}^{k} a_i \vec{b}_i$$

Kde $\vec{x}$ představuje aproximovaný výřez, a_i jednotlivé koeficienty a $\vec{b}_i$ bázové vektory. Počet těchto vektorů musí být větší nebo roven velikost aproximovaného výřezu. Pro většinu nalezených vektorů se koeficienty a_i blíží nule.

Pro automatizované nalezení bázových vektorů řídkého kódování lze využít speciální typy neuronových sítí označované jako autoenkodéry (autoencoder). Tato síť je učena na funkci identity – na vstup je pokládán výřez z obrazu a cílem je získat totožný výřez jako na vstupu. Architektura tedy odpovídá dopředné neuronové síti, přičemž vstupní a výstupní vrstva je stejně velká. Řídká aktivace je realizována omezením průměrné aktivace neuronů. Skrytou vrstvu sítě tedy neurony a váhy k nim vedoucí lze interpretovat jako řídké kódování (Ng, et al., 2016). Je-li ve skryté vrstvě méně neuronu než na vrstvě vstupní, hledá síť během učení úspornou reprezentaci obrazu. Lze ji tedy fakticky považovat za kompresi konkrétního výřezu. Díky tomu obsahuje taková reprezentace méně šumu a lze ji tak použít pro předzpracování obrazu.

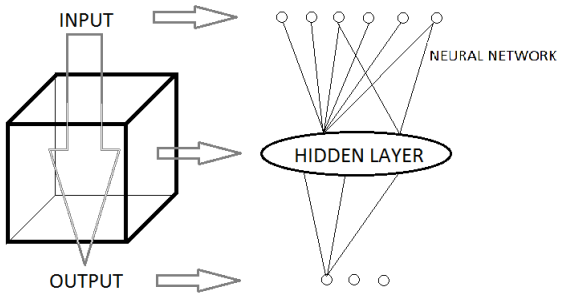


Obrázek 6: Architektura autoenkodéru a výsledné atributy nalezené pomocí řídkého kódování (obrázky volně převzaty z (Ng, et al., 2016))

Autoenkodéry je možné řetězit, kdy výsledná hierarchie zachycuje složitější struktury v obraze založené na kombinaci jednotlivých atributů.

5.6. Klasifikace

Klasifikace objektu spočívá v zařazení neznámého objektu o do konkrétní třídy c na základě společných rysů, obvykle vyjádřených jako vektor měření x provedených na objektu o. Třída c je potom popsána pravděpodobnostním modelem rozložení vektoru měření (Hendl, 2009) nebo reprezentativní množinou dat. Zařazení objektu do třídy provádí klasifikátor, na základě podobnosti posuzovaného objektu a modelu, s kterým klasifikátor pracuje. Ve zjednodušené podobě je možné klasifikátor chápat jako černou skříňku, jak je ilustrováno v Obrázek 7.



Obrázek 7: Analogie mezi klasifikátorem (black box) a neuronovou sítí

Klasifikátor je tedy model, který mapuje množinu vstupů na množinu výstupů. Tento model je obvykle založen na předem známé množině vstupů (učební množině). Vlastnosti modelu jsou pak obvykle posuzovány na základě schopnosti modelu klasifikovat předem neznámou množinu objektů (testovací množinu). Ačkoliv v ideálním případě je testovací množina klasifikována bez chyby, není to příliš obvyklé. Pro posuzování vlastností modelu jsou tedy často stanovována druhotná kritéria, v závislosti na podstatě problému, obvykle související s citlivostí klasifikace.

Klasifikace obrazu je obvykle založena na příznakovém popisu objektů v něm obsažených. Jako příznak se pak označuje elementární charakteristika popisovaného objektu, kterou lze

vyjádřit číselně (příznaky, a metody jejich získání jsou popsány v kapitole 5.5). Existují však také metody schopné klasifikace předzpracovaného obrazu bez nutnosti příznakového popisu. Většina takových metod využívá rozličných modelů umělých neuronových sítí, včetně výše zmíněných Yang, et al. (2002). Velký potenciál mají také konvoluční neuronové sítě a techniky obecně označované jako deep neural networks, které byly úspěšně použity pro rozpoznání mnoha tříd objektů, včetně ručně psaného písma, či dopravních značek. Referenční studie v zemědělských aplikacích však dosud chybí.

Následující přehled obsahuje klasifikační metody, které byly použity pro klasifikaci příznakových popisů rostlin v zemědělství a jejich základní princip.

5.6.1. Rozpoznání na základě minimální vzdálenosti

Je-li objekt popsán vektorem příznaků, lze příslušnost ke konkrétní třídě posoudit na základě vzdálenosti mezi modelem a objektem. K tomu slouží metody pro výpočet vzdáleností, dané sumou rozdílů prvků vektoru příznaků. Jako typický příklad lze uvést Hammingovu (26) vzdálenost či Euklidovu vzdálenost (27):

$$d_H = \sum_{i=0}^{n} |A_i - B_i| , \quad (31)$$

$$d_E = \sqrt{\sum_{i=0}^{n} (A_i - B_i)^2} \quad (32)$$

V níž jsou *A* a *B* odlišné vektory se stejným počtem složek. Objekt je potom přiřazen k třídě, ke které má minimální vzdálenost. Každá třída je pak obvykle reprezentována tzv. těžištěm, tedy vektorem, který reprezentuje průměrné hodnoty jednotlivých složek vektorů dané třídy. Hlavní nevýhoda takového přístupu je implicitní předpoklad stejné významnosti všech pozorovaných příznaků. Pro použití takovéto klasifikační metody by tedy musely být složky klasifikovaných vektorů normalizovány tak, aby jejich absolutní velikost odpovídala významnosti daného příznaku pro klasifikaci.

5.6.2. Diskriminační analýza

Diskriminační analýza je metoda vícerozměrné statistické analýzy, která slouží k řešení klasifikační úlohy, tj. zařazení objektů do existujících disjunktních tříd. Tato technika slouží k

nalezení odlišností ve stanovené populaci na základě mnohorozměrných dat a jejího rozdělení do rozlišitelných skupin. Odlišnosti v datech jsou popsány pomocí tzv. *klasifikačních rovnic*, které slouží ke klasifikaci objektů do skupin (podmnožin původní populace) pomocí hodnot vybraných proměnných, které popisují vlastnosti objektu. Tyto proměnné jsou kvantitativního charakteru a je předpokládáno jejich normální rozdělení v jednotlivých skupinách.

Diskriminační analýza vyjadřuje (Hendl, 2009):

- Do jaké míry jsou třídy separované pomocí použitých proměnných
- Jaké proměnné použít, aby byl objekt *o* zařazen do správné třídy *c*
- Jak vybrat optimální množinu proměnných

Existuje celá řada variant diskriminační analýzy, lišícími se jak principem odhadu parametrů, tak i tvarem diskriminační rovnice. Základním a nejjednodušším typem je potom lineární diskriminační analýza, která spočívá v popisu každé třídy c pomocí klasifikační rovnice ve tvaru:

$$L_c(x) = a_{1c}x_1 + a_{2c}x_2 + \ldots + a_{kc}x_k + a_{0c} \quad (33)$$

Při samotné klasifikaci je potom neznámý objekt popsaný pomocí vektoru *x* předložen klasifikačním rovnicím všech tříd a spočtena hodnota všech lineárních diskriminačních funkcí (diskriminační skóre). Objekt je přiřazen do třídy, na základě prahových bodů. Chybu klasifikace lze odhadnout pomocí Mahalanobisovy vzdálenosti. Tato klasifikační funkce je použitelná v případě, že se jednotlivé třídy liší pouze středními hodnotami tříd a nabývají normálního rozdělení.

Pro stanovení diskriminační rovnice je potom nutné vypočíst vektor koeficientů $\boldsymbol{\alpha}^T$ jako:

$$\boldsymbol{\alpha}^T = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{C^{-1}}, \quad (34)$$

Kde $\boldsymbol{\mu}_1$ a $\boldsymbol{\mu}_2$ jsou vektory průměrných hodnot příznaků pro každou z kategorií a $\boldsymbol{C}$ je sloučená kovariační matice. Konstantní člen je potom vyjádřen jako:

$$a_{0c} = -0.5\,\boldsymbol{\alpha}^T(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) - \ln(\pi_2/\pi_1), \quad (35)$$

Kde π_1 a π_2 jsou apriorní pravděpodobnosti výskytu jednotlivých skupin.

Výsledky diskriminační analýzy jsou závislé na řadě faktorů. Především pak hraje roli (Meloun & Militký, 2004):

- Volba nezávislých znaků (diskriminátorů) a závislých proměnných
- Poměr velikosti výběru a počet diskriminátorů
- Dělení výběru ke klasifikačním účelům

Pro úspěšnou aplikaci je také předpokládána vícerozměrná normalita diskriminátorů a absence multikolinearity.

5.6.3. Rozhodovací stromy

Rozhodovací stromy (decision trees) jsou datové struktury stromového typu hojně používané v operačním výzkumu, využitelné pro klasifikaci i řešení regresních úloh. V případě klasifikačních úloh představuje každý uzel stromu jeden z atributů (vlastností, příznaků) určovaného objektu, na jehož základě lze objekty rozlišit do tříd. Z každého uzlu vychází konečné množství hran. Listy stromu pak představují výsledné třídy objektů, přičemž třída může být reprezentována více než jedním listem, k nimž lze dospět různými cestami. Jednotlivé jsou také často ohodnoceny, díky čemuž lze vyhodnotit pravděpodobnost výsledné klasifikace. Pro klasifikaci objektů jsou často používány tzv. binární stromy, kdy z každého uzlu vychází právě 2 hrany.

Pro sestavování rozhodovacích stromů jsou používány rekurentní algoritmy, které dle řadí jednotlivé atributy dle jejich míry informační hodnoty (tedy na základě toho, jak efektivně rozlišují vstupy od podkategorií) a následně je hierarchicky skládá do stromu. Častými optimalizace mi je pak prořezávání stromu (angl. pruning) a Známé jsou zejména algoritmy ID3 (Quinlan, 1986) a jeho rozšíření C4.5 (Quinlan, 1993). Algoritmus

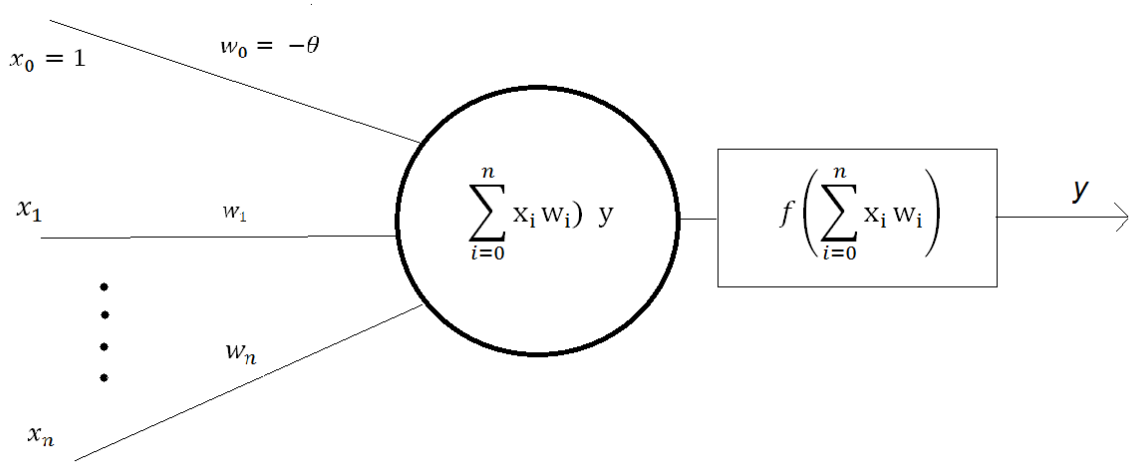
Předpokládejme, že každý z objektů je popsaný množinou atributů A. Obecný přístup pro tvorbu rozhodovací stromu je (Quinlan, 1993):

- Pro každý atribut $a_i \in A$ je vypočtena normalizovaná míra informační základě rozdělení množiny vstupů podle atributu a_i.
- Nechť je *a_best* atribut s nejvyšší normalizovanou mírou informační hodnot. Na jeho základě je v grafu vytvořen nový uzel, který rozdělí množinu vstupů do podtříd
- Algoritmus je opakován na takto vzniklých potomcích
- Patří-li všechny vzorky do jedné třídy, vzniká list. Neposkytují-li další atributy informační hodnotu, algoritmus pokračuje na uzlech výše v hierarchii stromu.

Výhodou rozhodovacích stromů je jejich snadná interpretace z pohledu člověka a rychlá kvalifikace. Rozšířeným problémem je naopak přetrénování, resp. návrh rozhodovacích stromů na základě omezené učební množiny, kde je některá z použitých tříd použita častěji. Takové stromy nejsou schopny generalizace a jejich využitelnost pro data neobsažená v učební množině je omezená.

5.6.4. Neuronové sítě

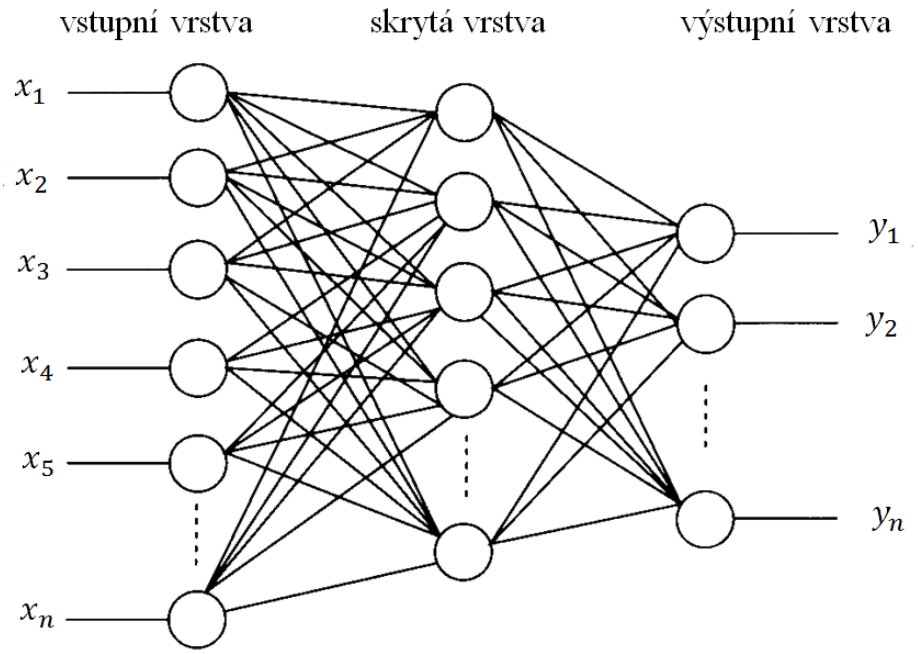
Umělá neuronová síť je výpočetní model umělé inteligence inspirovaný centrálním nervovým systémem vyšších organizmů. Skládá se z jednotlivých výpočetních jednotek, tzv. neuronů, které jsou vzájemně propojeny, a navzájem si předávají signály. Zobecněný matematický model neuronu s výstupní aktivační funkcí f je schematicky vyobrazen na Obrázek 8. Neuron se sigmoidální (logistickou) aktivační funkcí je poté schopen klasifikovat lineárně separabilní množiny $X \subset \mathrm{E}^n$ s nulovou chybou.



Obrázek 8: Matematický model neuronu

Při řešení klasifikační úlohy je potom cílem najít takovou konfiguraci vah neuronu, která klasifikuje neurony s nejmenší možnou chybou. Tento proces je označován jako učení neuronu. Učení je realizováno prostřednictvím iterativního algoritmu na základě učebního souboru D který je tvořen množinou dvojic (x, d), kde d je kód kategorie, do které vstupní vektor x patří. Prvky z učební množiny se postupně předkládají neuronu, a na jejich základě se mění váhy neuronu po každém vstupu (iterativní či online algoritmy) nebo po jedné trénovací epoše (bakchové algoritmy). Učení pokračuje, dokud není dosazeno ukončující podmínky. Touto je obvykle dosažení klasifikační chyby menší než předem stanovená hodnota, nebo maximální počet iterací.

Vícevrstvá perceptronová síť (také *Multi-Layer Perceptron* či zkráceně MLP) je acyklická dopředná neuronová síť. První vrstva je považována za vstupní, poslední vrstva za výstupní a veškeré vrstvy mezi nimi jsou označovány jako vrstvy skryté. Propojené jsou vždy jen vrstvy sousední, přičemž propojení v rámci sítě je úplné, tedy každý neuron je propojený se všemi neurony v následující vrstvě. Každé propojení je pak ohodnoceno vahou. Schematický model vícevrstvé sítě je ilustrován na Obrázek 9.



Obrázek 9: Neuronová síť (MLP) s jednou skrytou vrstvou

Neuronová síť má schopnost generalizace, tedy schopnost úspěšně klasifikovat velkou část případů, které nebyly součástí učební množiny. Tato vlastnost je jedním z nejdůležitějších předpokladů pro úspěšnou aplikaci neuronových sítí při řešení praktických problémů, proto je nutné schopnost generalizace během učení pravidelně testovat. Toto je realizováno výpočtem chyby po stanoveném počtu učebních epoch, a porovnání její hodnoty na učebním a validačním souboru. V momentě, kdy začne chyba na validačním souboru stoupat, zatímco na učebním souboru klesá (moment přeučení sítě), je učení nutné zastavit. Náchylnost sítě k přeučení lze snížit regularizací či snížením počtu neuronů ve skryté vrstvě.

Nejpoužívanějším algoritmem pro učení neuronové sítě je algoritmus zpětného šíření chyby (backpropagation of error). Algoritmus backpropagation of error pro klasifikační úlohy funguje následujícím způsobem:

- Je stanoven počáteční vektor vah w^0 (nejčastěji jsou jeho složky náhodně generovány z intervalu (-1;1)).

- Prvky učebního jsou postupně předkládány síti. Předpokládejme, že v n-tém kroku je vybrán prvek x^n. Potom

(36) $w^n = w^{n-1} - \varepsilon\, gradE + \mu\Delta^n$

Kde $\varepsilon > 0$ je velikost učebního kroku, $\mu > 0$ je tzv. moment, který ovlivňuje rychlost a stabilitu učebního algoritmu a parametr $\Delta^n = w^{n-1} - w^{n-2}$ je daný poslední změnou váhového vektoru. Jako chybová funkce při klasifikaci je pak pro m kategorií obvykle používána funkce:

(37) $E = -\sum_x \sum_{i=1}^{m} d_i \log Y_i$

Algoritmus končí, jakmile je dosaženo ukončující podmínky, obvykle dané hodnotou chybové funkce, nebo maximálním počtem iterací.

Od vytvoření algoritmu backpropagation of error vznikla celá řada variací na tento úspěšný algoritmus a heuristik, které učení neuronové sítě urychlují. Typickým zástupcem takových heuristik je například algoritmus *resilient backpropagation* (Riedmiller & Braun, 1992), který na rozdíl od původního algoritmu neřeší absolutní velikost změny gradientu, pouze jeho směr.

Řešení klasifikační úlohy pomocí neuronové sítě je obecně realizováno v následujících krocích:

1. Jasná formulace klasifikační úlohy
2. Analýza datového souboru, včetně ošetření chybějících datových záznamů, normalizace dat a případné redukce dimenze vstupních dat
3. Rozdělení datové souboru na učební (trénovací), validační a testovací množinu
4. Volba architektury sítě
    a. Velikost vstupní a výstupní vrstvy je určena charakterem úlohy
    b. Počet skrytých vrstev a neuronů v nich obsažených
    c. Typy neuronů ve skryté a výstupní vrstvě, resp. typ jejich aktivační funkce
    d. Chybová funkce
    e. Počáteční nastavení vah

5. Volba algoritmu učení, včetně parametrů (krok, moment) a ukončující podmínky
6. Výběr výsledné sítě včetně hodnocení chyby na testovacím souboru, a stanovení intervalů spolehlivosti

Hlavním problém je tedy volba vhodné architektury sítě a algoritmu učení včetně jeho parametrů. Ačkoliv se tímto problémem zabývala celá řada výzkumů, jednoznačná odpověď, jak zvolit architekturu a počáteční konfiguraci sítě dosud chybí. Zajímavou tematické studii na toto téma předložili např. Keeni et al. (1999) zaměřující se na příznakové klasifikace; Sheela a Deepa (2013) naproti tomu v přehledové části své studie předkládají přehled obsahující nejpoužívanější heuristiky či tzv. rule of thumb. Volba architektury sítě je tak ve většině případů dosud vybírána experimentálně.

5.6.5. Random forests

Random forests (také random decision forests, náhodné lesy) (Breiman, 2001) je metoda využitelná pro klasifikaci objektů. Tato metoda vytváří během učení řadu množinu M rozhodovacích stromů $P_1,...,P_M$. Základními myšlenkami pro tvorbu random forests jsou zásadními náhodný výběr atributů založený na metodě náhodných podprostorů (Ho, 1995), a bagging (Breiman, 2001), tedy metodu náhodného výběru k podmnožin z trénovacího souboru. V případě random forests není kladen důraz na kvalitu jednotlivých stromů – cílem je minimalizovat chybu rozhodování celého lesa. Velikost růstu stromů není omezena, prořezávání v základní variantě také aplikováno není. Důležitými parametry rozhodovacích stromů jsou počet vzniklých stromů (k) a počet vybraných proměnných, které jsou použity pro rozdělení stromu na jednotlivé uzly (m) (Peters, et al., 2007). Tyto parametry jsou optimalizovány tak, aby byla obecná chyba klasifikace minimální.

Oproti klasickým rozhodovacím stromům jsou random forests robustnější vůči šumu a méně náchylné k přeučení (Breiman, 2001). Jejich doba učení je ale výrazně delší (je sestaveno několik stromů) a také rozhodování o jednotlivých případech trvá déle, neboť rozhodnutí je vypočteno pro každý strom samostatně a následně musí být hlasováním stanoveno konečné rozhodnutí.

5.6.6. Boosting

Boostingové algoritmy vycházejí z hypotézy, že množina slabých klasifikátorů (s přesností mírně přesahujících náhodnou volbu) mohou dohromady vytvořit silný klasifikátor (odhadovaný výsledek koreluje se skutečným výstupem).

Zřejmě nejčastěji používaným příkladem takového algoritmu je AdaBoost (zkr. Adaptive Boosting) (Freund, 2001). Boostovaný klasifikátor nabývá formy:

$$F_T(x) = fn \sum_{t=1}^{T} \alpha_t f_t(x) \,, \tag{38}$$

Kde každé f_t je slabý klasifikátor do něhož vstupuje objekt *x* jako vstup a jehož výsledkem je hypotéza o příslušnosti objektu x ke konkrétní třídě, obvykle doplněná mírou spolehlivosti či pravděpodobností klasifikace a α_t značí váhy jednotlivých slabých klasifikátorů. Výsledná klasifikace je dále realizována pomocí funkce *fn* (nejčastěji *sgn* pro případy detekce, či softmax při klasifikaci).

Postup učení výsledného klasifikátoru je sekvencí 3 základních kroků. Při každé iteraci *t* je vybrán slabý klasifikátor s nejmenší váženou chybou v predikci na základě minimalizace chyby E_t, dle:

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t h(x_i)] \,, \tag{39}$$

Kde $F_{t-1}(x_i)$ je boostovaný klasifikátor vzniklý v předchozí fázi učení, E je chybová funkce a $\alpha_t h(x_i)$ popisuje chybový klasifikátor, který je testován jako další součást výsledného klasifikátoru. Následně je nově nalezenému klasifikátoru přiřazena váha na základě počtu správných klasifikací.

Při každé iteraci učebního procesu jsou také přiřazovány váhy jednotlivým vzorům v učební množině na základě chyby $E(F_{t-1}(x_i))$ pro daný vzor. Na základě těchto vah lze zvolit množinu případů s vysokou chybou a najít slabý klasifikátor, který tyto případy správně rozlišuje. Tento klasifikátor je potom přidán do výsledného boostovaného klasifikátoru. Výsledkem je lineární kombinace jednotlivých klasifikátorů, která minimalizuje klasifikační chybu.

5.7. Konvoluční neuronové sítě (CNN)

Konvoluční neuronové sítě (také *hluboké konvoluční sítě*) jsou rozšířeným modelem dopředných neuronových sítí (obvykle hlubokých) s architekturou uzpůsobenou pro řešení úloh klasifikace obrazu. Jejich architektura volně vychází ze Fukushimovy sítě *Neocognitron* (Fukushima, 1980). První moderní a všeobecně rozšířenou variantou je pak síť LeNet použitá pro klasifikaci písmen z databáze MNIST.

Základním principem je aplikace několika vrstev filtrů (konvolučních jader) na zdrojové obrazové funkce, které slouží k výběru příznaků. Nejsilnější lokální příznaky jsou obvykle po každé konvoluční vrstvě vybírány pomocí podvzorkovacích vrstev. Zpracování obrazu oběma typy vrstev vede k redukci prostorové informace obrazu. Hlubší vrstvy vybírají příznaky vyšší úrovně. Na konci extrakce je vektor hodnot, který je použit jako vstup pro dopřednou (plně propojenou) neuronovou síť. Na základě tohoto příznakového prostoru je objekt v obrazu rozpoznán.

Tyto typy sítí jsou primárně používány pro klasifikaci obrazů (co je na obrázku), shlukování obrazů na základě jejich podobnosti (vyhledávání obrazů) a rozpoznávání objektů ve scéně. Rozličné varianty konvolučních neuronových sítí tak byly použity např. pro identifikaci tváří, dopravních značek (Cires¸an, et al., 2011) či klasifikaci obrázků z databáze CIFAR (Ciregan, et al., 2012).

5.7.1. Konstrukční prvky konvoluční neuronové sítě

Konvoluční neuronová síť se skládá ze dvou základních vrstev, které slouží k popisu a redukci příznakového prostoru:

- Vrstva konvoluční
- Vrstva sdružování (také podvzorkování)

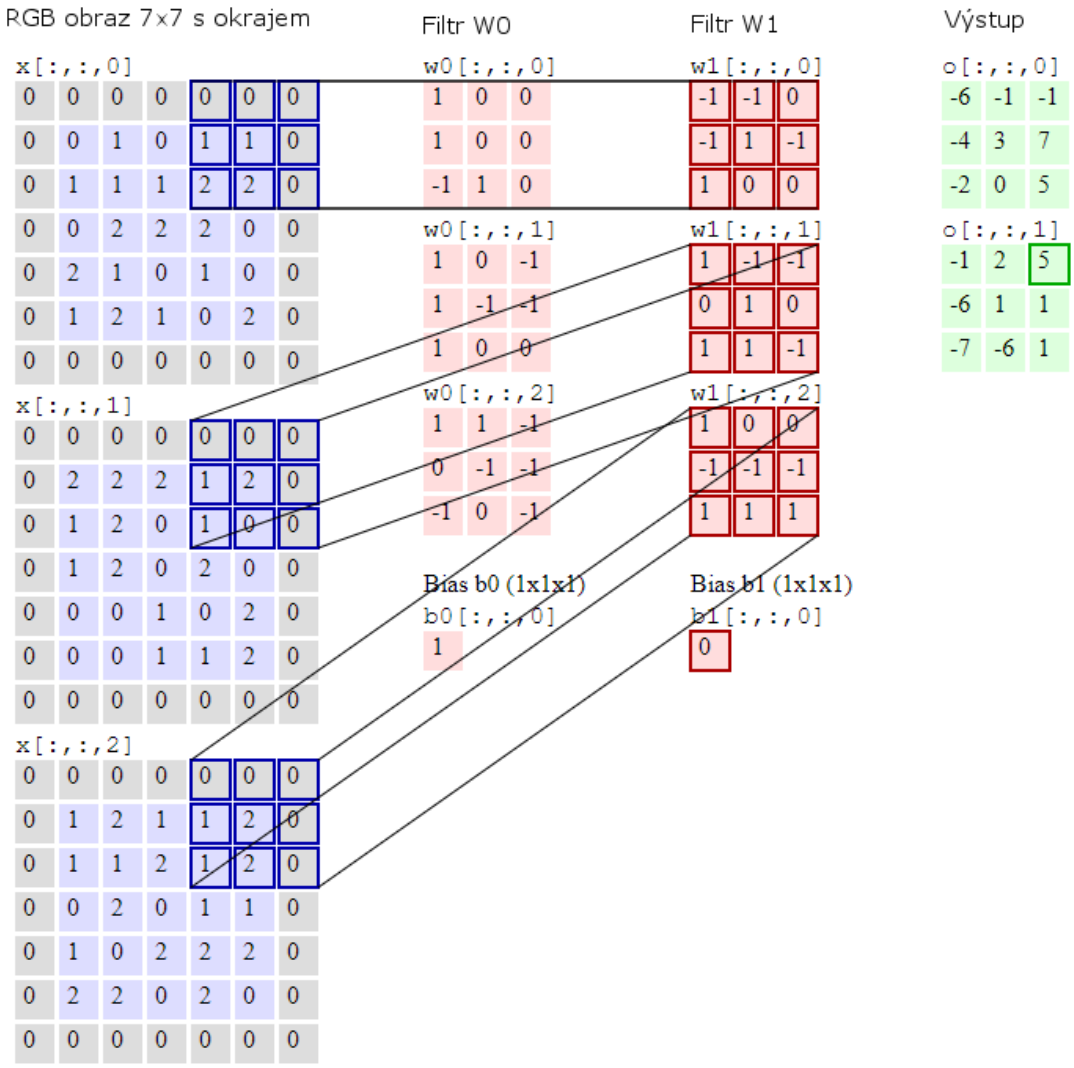
Tyto dva typy vrstev jsou využívány prakticky všech architekturách konvolučních sítí. Tyto vrstvy jsou obvykle doplněny dalšími určenými především pro zrychlení učení modelu a zabránění přeučení. Při tvorbě jednotlivých modelů, které budou dále představeny však vznikla

i celá řada dalších podpůrných řešení a principů. Tato kapitola vysvětlí principy těch nejdůležitějších konstruktů.

5.7.1.1. Konvoluční vrstva

Konvoluční vrstva je založena na myšlence výpočtu lokálních charakteristik obrazu pomocí konvolučních jader (filtrů). Vstupem konvoluční vrstvy je obraz o rozměrech $m \times n \times r$, kde m je výška a šířka a r je počet kanálů v obrázku, tedy pro RGB je r=3. Konvoluční vrstva obsahuje k filtrů (konvolučních jader) o rozměrech $n \times n \times q$, kde n je velikost jádra menší než menší strana obrazu, a q může být stejné nebo menší jako počet kanálů v obraze. Velikostí filtrů je dána velikostí lokálně propojených struktur, které vytváří příznakové mapy o velikosti $m–n + 1$, je-li velikost posunu jádra po obrazu 1.

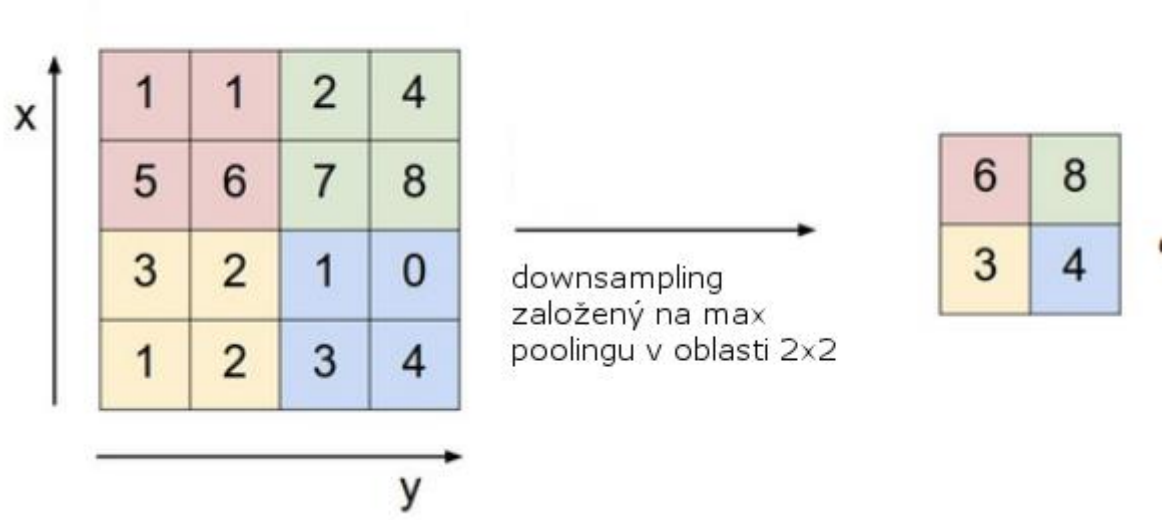
Každá konvoluční vrstva mění reprezentaci obrazové funkce tak, aby byly zvýrazněny nejdůležitější oblasti obrazu (Obrázek 10). To vede k celkové redukci příznakového prostoru.



Obrázek 10: Princip konvoluční vrstvy [animace: (Karpathy, 2017)]

5.7.1.2. Sdružovací (podvzorkovací) vrstva

Sdružovací vrstva slouží především k nelineárnímu podvzorkování signálu (downsampling). Tato vrstva počítá lokální statistické údaje z předchozí konvoluční vrstvy pomocí funkcí typu max/mean. Nejčastěji používaná je pak varianta označovaná jak max pooling, která vybírá pro další zpracování oblast, v nichž je vzor signálu nejsilnější (Obrázek 11: Princip vrstvy podvzorkování (max pooling) Obrázek 11). Díky tomu jsou vybrány lokálně významné a kvalitní příznaky.



Obrázek 11: Princip vrstvy podvzorkování (max pooling) (Karpathy, 2017)

Slučování vede k možnosti hledat příznaky vyšší úrovně v dalších vrstvách sítě, přičemž rozměry konvolučních jader zůstávají zachovány. Tato úprava také zrychluje celkovou relaxaci sítě.

5.7.1.3. Aktivační funkce ReLU

Zatímco dopředné neuronové sítě běžně využívají varianty logistické aktivační funkce, konvoluční sítě využívají jednoduchou variantu označovanou jako ReLU (Hahnloser, et al., 2000). Tato funkce je reprezentována v základně variantě předpisem:

$$f(x) = \begin{cases} x, & je - li\ x \geq 0; \\ 0, & je - li\ x < 0. \end{cases} \tag{40}$$

ReLU aktivace je nejjednodušší nelineární aktivační funkcí, kterou je možné použít. Její hlavní výhodou je rychlost při učení v rozsáhlých sítích. Tato aktivační funkce má samozřejmě také nevýhody – při použití algoritmů učení využívající zpětný tok gradientů může neuron dospět ke stavu, kdy zůstává neaktivním pro libovolné vstupy, zejména je-li míra učení příliš vysoká. Pro snížení pravděpodobnosti výskytu tohoto problému se často využívá metoda označovaná jako Leaky ReLU, která nabývá nenulové hodnoty dle základního předpisu:

(41) $$f(x) = \begin{cases} x, & je - li\ x \geq 0; \\ 0.01x, & je - li\ x < 0. \end{cases}$$

Na podobném principu pracuje také funkce ELU (exponenciální lineární jednotka), která se snaží přiblížit průměrnou hodnotu aktivace blíže k nule a zrychlit tak učení. Je vyjádřena předpisem:

(42) $$f(x) = \begin{cases} x, & je - li\ x \geq 0; \\ a(e^x - 1), & je - li\ x < 0. \end{cases}'$$

kde *a* je hyperparametrem sítě, který nabývá pozitivních hodnot.

5.7.1.4. Optimalizace algoritmů gradient descent

Při učení neuronových sítí jsou obvykle využívány varianty algoritmů gradient descent (tedy „klesání podle gradientu“). V souvislosti s těmito algoritmy vyvstává řada problémů, které jsou dále zvýrazněny počtem a podobou vrstev v konvolučních neuronových sítích. Nejvýznamnější z nich pak zahrnují například:

- Volbu správné hodnoty parametrů learning rate (míra učení)
- Volbu a přípravu learning schedules (učební rozpisů), jejichž hlavním cílem je korigovat hodnoty parametru learning rate během učení
- Parametr learning rate je využíván pro všechny parametry stejným způsobem, přestože by často bylo výhodnější významněji zdůraznit vlastnosti, které se vyskytují řídce
- Problém uváznutí v lokálních minimech

V souvislosti s konvolučními neuronovými sítěmi jsou často využívány specifické optimalizační algoritmy, které slouží k řešení těchto problémů. Mezi nejčastěji používané algoritmy v současných implementacích konvolučních neuronových sítí patří AdaGrad, RmsProp, či ADAM, které korigují hodnotu míry učební pomocí členu označovaného souhrnně jako momentum.

Algoritmus AdaGrad využívá předchozí hodnoty gradientu pro daný parametr, a upravuje pomocí něho hodnoty parametru learning rate (α). Toto je realizováno pomocí podílu současné

hodnoty gradientu a součtu gradientů. Díky tomu je platí, že je-li hodnota gradientu vysoká, je míra učení snížena a naopak. Toto kritérium lze vyjádřit jako:

$$(43) \quad \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t}+\varepsilon}\ g_t,$$

kde θ_t je váha v čase t, $G_t \in \mathrm{R}^{dxd}$ je diagonální matice, kde každý diagonální prvek je součet čtverců gradientů vztažených k váze θ_t do časového bodu t, g_t gradient v čase t a ε parametr, který zabraňuje dělení nulovou (obvykle velmi malé hodnoty, např. 1e-8). Tato metoda vede k postupnému poklesu míry učení, která může být v mnoha případech nežádoucí.

Podobným případem je metoda RMSProp, která vychází z výše uvedené metody AdaGrad, avšak používá exponenciální redukční tvar míry učení dle předpisu:

$$(44) \quad \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t}+\varepsilon}\ .\ g_t\ ;\ E[g^2]_t = 0.9\ E[g^2]_{t-1} + 0.1 g_t^2$$

Algoritmus Adaptive Moment Estimation (ADAM) (Kingma & Ba, 2015) dále rozšiřuje RMSprop o bias-korekci a momentum. Tento algoritmus zakládá korekci míry učení na postupně klesajících momentech, které jsou dále korigovány pomocí parametrů parametry bias korekce β_1^t *a* β_2^t . Aktualizaci vah pak lze zapsat předpisem:

$$(45) \quad \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t}+\varepsilon}\ .\hat{m}_t\ ;\ \hat{m}_t = \frac{m_t}{1-\beta_1^t}\ ;\ \hat{v}_t = \frac{v_t}{1-\beta_2^t},$$

kde $\hat{m}_t$ je korigovaný první moment (průměr) a $\hat{v}_t$ druhý moment (rozptyl) gradientů za předchozí období. V současnosti je algoritmus ADAM zřejmě nejpoužívanějším z této skupiny.

5.7.1.5. Normalizace dávky (Batch Normalization)

Batch Normalization je vrstva konvoluční neuronové sítě využívaná v procesu extrakce příznaků, která během učení průběžně vyhodnocuje plovoucí průměr a rozptyl pro výstup předcházející vrstvy. Pomocí těchto statistických ukazatelů jsou potom výstupy před vstupem do další vrstvy normalizovány tak, že je jejich střední hodnota nulová. Díky normalizaci je rozdělení hodnot stabilnější, což vede k rychlejší konvergenci gradientních optimalizačních metod (Ioffe & Szegedy, 2015), a tedy k vyšší rychlosti trénování.

Normalizace může ovlivnit schopnost sítě generalizovat. Vrstva normalizace dávky proto zavádí 2 parametry, měřítko (γ) a posun (β). Normalizace je pak realizována jako:

$$\vec{x} = \frac{\vec{x} - (mean'[x] - \beta)}{\sqrt{var'[x]}}\gamma \quad (46)$$

kde $mean'[x]$ je plovoucí průměr a $var'[x]$ plovoucí rozptyl. Měřítko i posun je možné trénovat jako váhy v tradičních vrstvách.

5.7.1.6. Regularizace neuronových sítí

Učení neuronových sítí běžně pracuje s obrovským množstvím parametrů. V případě konvolučních sítí jsou pak těchto parametrů běžně desítky miliónů. V důsledku toho často dochází k přetrénování sítě – tedy přílišné adaptaci na učební množinu a současnou ztrátu schopnosti generalizovat projevují se velkou chybou na testovací sadě. Hlavním cílem regularizace je omezit výskyt tohoto problému. Některé metody regularizace taky vedou ke zvýšení rychlosti učení.

Nejběžnější skupina regularizačních metod je založena na penalizace velikosti vah pomocí přidání specifického regularizačního kritéria *K*, které je přidáno k funkci, která je použita při učení, dle obecného předpisu:

$$f(w) = \sum_{i=1}^{n} error(N(w, x_i), y_i) + \lambda K \; ; \; argmin\, f(w) \quad (47)$$

kde λ je parametr, který ovlivňuje, jak významný je vliv kritéria K, tedy sílu regularizace. Díky regularizačním kritériím jsou omezeny extrémní aktivace neuronů a drobná změna ve vstupu tak nevede k prudkým změnám výstupu. Při špatné volbě regularizačního parametru λ by však síť mohla ztrácet schopnost správně rozlišovat mezi kategoriemi.

L2 je zřejmě nejčastěji používanou metodou regularizace pro konvoluční neuronové sítě. Regularizační kritérium K je zde definováno jako: $\sum_{i=1}^{n} w_i^2$. Regularizace L2 nejsilněji penalizuje nejvyšší hodnoty vah, což vede k tomu, že síť částečně využívá veškeré vstupy, které jsou ji poskytovány, namísto přehnaného důrazu na několik vysokých vah. Výsledkem je to, že

je reprezentace úlohy rozptýlena napříč neurony v úloze, přičemž váhy nabývají poměrně nízkých hodnot.

Metoda regularizace označovaná nejčastěji jako L1 (také Lasso regression) využívá regularizační kritérium ve tvaru $\sum_{i=1}^{n} |w_i|$. Při použití regularizace L1 se hodnota mnoha vah blíží k nule, což fakticky vede k tomu, že se síť ve výsledku stává řídce propojenou. Tato metoda regularizace tak může být považována za metodu sloužící k výběru příznakového podprostoru a výsledná síť se stává méně citlivou vůči šumům. Není-li však cílem regularizace explicitní redukce příznakového prostoru, je za vhodnější považována metoda L2.

Regularizace L1 a L2 jsou také často kombinovány v rámci tzv. Elastic net regularizace, která využívá kritérium ve tvaru $\lambda_1 \sum_{i=1}^{n} |w_i| + \lambda_2 \sum_{i=1}^{n} w_i^2$.

Často využívanou metodou regularizace je také učení sítí na různých (často náhodných) částech učební množiny a následný vyhodnocení klasifikace na základě většinového hlasování všech sítí, resp. výpočtu průměrné pravděpodobnosti. Díky takovémuto mechanismu je celková klasifikace odolnější vůči přetrénování – ačkoliv jednotlivé sítě mohou být přetrénovány, výsledek je založen na klasifikaci všech sítí, přičemž je stále zastoupena kompletní učební množina. Díky oddělenému trénování jsou však sítě navzájem nezávislé, a výsledný klasifikátor tedy lépe zobecňuje. Nevýhodou takovéhoto přístupu jsou vysoké požadavky na výpočetní prostředky (musí být naučeno větší množství sítí) jakož i na velikost učební množiny. V případě klasifikace obrazu a objektů v něm obsažených toto může znamenat nutnost získat desítky tisíc klasifikovaných obrázků nemluvě o neúměrně rostoucí době učení. Z toho důvodu je tato metoda využívána poměrně vzácně.

Na analogickém logickém přístupu je založena regularizační metoda označovaná jako dropout. V tomto přístupu je síti v rámci každé učební epochy předložena pouze vybraná část učební množiny, přičemž vybrané procento neuronů v síti je po dobu učení neaktivních. Algoritmus lze stručně popsat jako:

1. Výběr podmnožiny M z učební množiny, tak, že platí $M \subset D$
2. Výběr n% neuronů (obvykle 50%) a jejich deaktivace
3. Učení sítě na vybrané sadě M

Hinton et al. (2012) dokázali, že jsou-li po dokončení učení veškeré váhy vyděleny 2, odpovídá výsledná klasifikace sítě geometrickému průměru všech dílčích sítí poloviční velikosti.

5.7.2. Transfer learning a retraining

V mnoha architekturách konvolučních neuronových sítích lze snadno odlišit části sloužící pro extrakci příznaků a části které realizuje samotnou klasifikaci. Je-li pro učení sítě využita rozsáhlá množina učebních obrazů lze úspěšně předpokládat, že mnohé charakteristické příznaky, které lze využít pro klasifikaci objektů (jako rohy, hrany aj.) již byly v extrakční části naučeny a nové objekty bude možné rozlišit pouze na základě odlišné kombinace těchto příznaků. V mnoha případech je skutečně možné znovu využít konvoluční a sdružovací vrstvy využité pro extrakci příznaků a na nich popsat dodatečné tříd vzorů. Toto je realizováno změnou struktury (přidáním neuronu do výstupní vrstvy) a přeučením výstupních klasifikačních vrstev (plně propojené vrstvy, softmax vrstva). Tato technika vede ke významnému zkrácení času učení sítě. Zároveň pak pro problémovou oblast vzniká rozsáhla množina již extrahovaných příznaků, které lze využívat i v dalších, příbuzných oblastech.

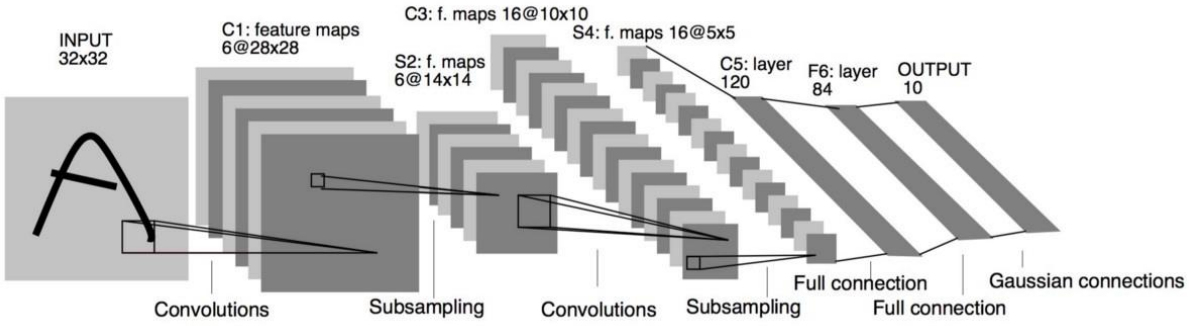
5.7.3. Význačné architektury CNN pro rozpoznání a detekci objektů v obrazu

Konvolučních neuronové sítě lze v současnosti považovat za nejdynamičtěji rozvíjející se oblast studia rozpoznání a klasifikace obrazu. V posledních deseti letech vznikla celá řada architektur založených na využití konvolučních vrstev. Tyto sítí postupně vytvářejí abstraktnější vektory atributu, které objekty zachycují ve formě, která je robustnější vůči různým deformacím. U všech níže uvedených vrstev roste s hloubkou počet použitých konvolučních jader, a tedy i velikost vektoru atributů. V naprosté většině případů je toto kompenzováno sdružovacími vrstvami, které snižují velikost vstupu.

5.7.3.1. LeNet5

Architektura LaNet (LeCun, et al., 1998) je síť navržená pro rozpoznání písmen, která přinesla celou řadu zásadních poznatků pro rozpoznání obrazu, zejména myšlenku, že příznaky jsou rozšířeny napříč celým obrazem a konvoluce si určitelnými parametry jsou efektivní způsob, jak extrahovat podobné funkce na více místech obrazu s několika parametry. LeNet5 vychází z

předpokladu, že obrázky jsou vysoce prostorově korelované a použití jednotlivých pixelů vede ke ztrátě této informace.



Obrázek 12: Architektura sítě LeNet5 [převzato z (LeCun, et al., 1998)]

V síti LeNet se poprvé objevily mnohé základní myšlenky, které jsou základem většiny následujícího výzkumu v oblasti CNN, včetně:

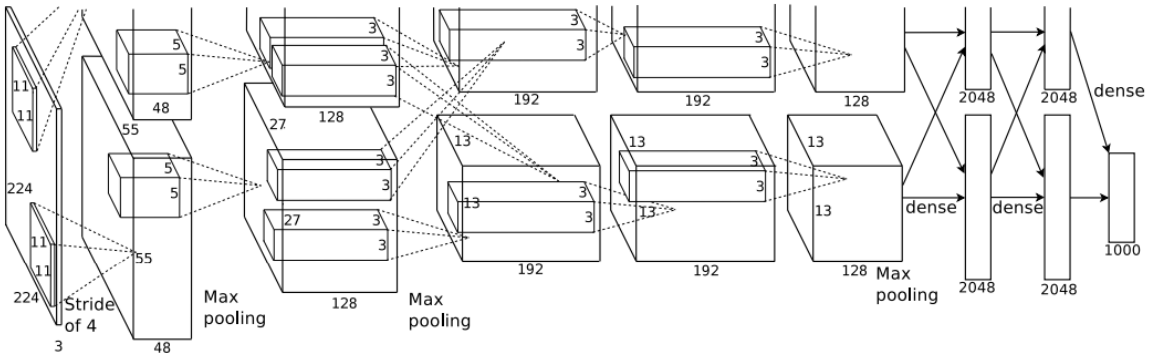
- Konvoluční neuronové sítě používají kombinaci 3 vrstev: konvoluce, sdružování, a non-linearity
- Konvoluce pro extrahování prostorových prvků
- Podvzorkování s využitím prostorového průměru příznakových map
- Non-linearita ve formě tanh/sigmoidální funkce
- Vícevrstvá neuronová síť realizuje závěrečnou klasifikaci příznakového vektoru
- Řídké připojení v konvoluční vrstvě, snižuje výpočetní náklady a zachovává prostorový vztah (korelaci) mezi body

Tato síť byla využívána ke klasifikaci ručně psaných znaků (dataset MNIST), tvoří však základ většiny současných modelů rozpoznání objektů.

5.7.3.2. AlexNet

Architektura AlexNet (Krizhevsky, et al., 2012) je určena pro rozpoznání obrázků, použitá originálně pro klasifikaci obrazové množinu ILSVRC-2010 do 1000 kategorií. Síť AlexNet přímo vychází ze architektury LeNet – je to její hlubší a širší verze, která je využitelná pro rozpoznání výrazně komplexnějších objektů. Architektura AlexNet ses skládá z pěti

konvolučních vrstev, z nichž některé jsou následovány sdružovací vrstvou (max pooling), a třemi plně propojenými dopřednými vrstvami z nichž závěrečná je realizována metodou softmax. o



Obrázek 13: Architektura sítě AlexNet (Krizhevsky, et al., 2012)

Mezi hlavní přínosy patří:

- Použití opravených lineárních jednotek (ReLU) jako Non-linearity
- Použití regularizace pomocí dropout techniky, což zabraňuje přeučení.
- Použití techniky rozšíření dat, včetně překladu obrázků, horizontálních odrazů, a extrakce výřezů z obrazu
- Překrývající se maximální sdružování (max pooling), čímž se zamezí průměrným vlivům průměrného sdružování

Síť AlexNet je také první síť, která v procesu učení výrazně využila GPU, což vedlo až k desetinásobnému zvýšení rychlosti učení a možnosti pracovat s většími datovými množinami a většími obrázky.

5.7.3.3. VGG

Architektura VGG (Simonyan & Zisserman, 2014) je určena pro rozpoznání obrazu (v testu ILSVRC 2014 dosáhla chybovosti 7.3%) a lokalizaci objektů v obrazu obsažených. Základní varianta této architektury se skládá z 19 vrstev, přičemž využívá výhradně konvoluční filtry 3x3 s krokem a okrajem o velikosti 1x1, a sdružovací 2x2 vrstvy (max pooling). Konvoluční

vrstvy 3x3 jsou naskládány za sebe, čímž je fakticky dosaženo efektivní oblasti vnímání 7x7, přičemž celkový počet parametrů sítě klesá.

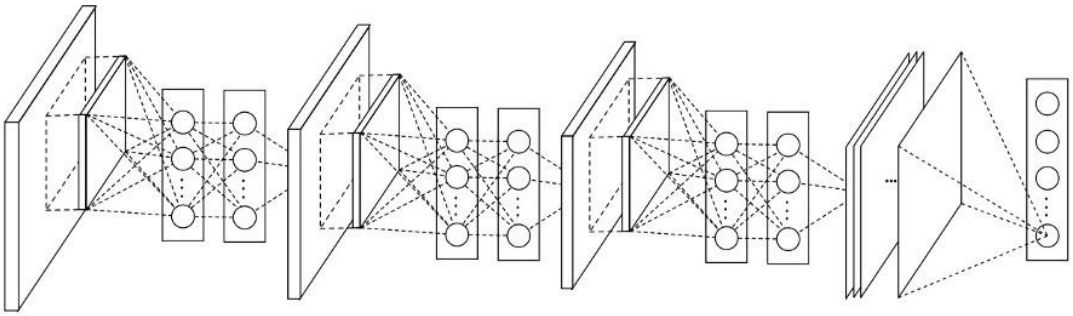
Nejdůležitější novinky v architektuře zahrnují:

- Použití navazujících konvolučních vrstev 3x3 nahrazuje větší jádra
- Po každé konvoluční vrstvě následuje ReLU vrstva
- Se zmenšujícími se prostorovými rozměry oblázku v důsledku konvolučních a sdružovacích vrstev roste počet příznakových map – po každé sdružovací vrstvě se počet použitých filtrů zdvojnásobuje
- Pro rozšíření učební množiny využívá kolísání měřítka (scale jittering)
- Pro učení využívá algoritmus batch gradient descent

Z principů použitých v architektuře VGG vychází celá řada dalších modelů architektur včetně SSMD (viz. níže) a architektur určených pro segmentaci obrazu.

5.7.3.4. Network-in-network

Architektura Network in Network (NiN) (Min, et al., 2013) je dalším typem architektury určené primárně pro rozpoznání obrázků. Hlavními přínosy této architektury je nahrazení běžných lineárních vícevrstvou perceptronovou sítí. Díky této úpravě dokáže síť lépe kombinovat složitější příznaky zachycené konvoluční operací. Další přínosem je pak využití konvoluce 1x1 napříč příznakovými mapami. Tato úprava dokáže efektivně zachytit prostorové vztahy mezi příznaky pomocí relativně malého množství parametrů.



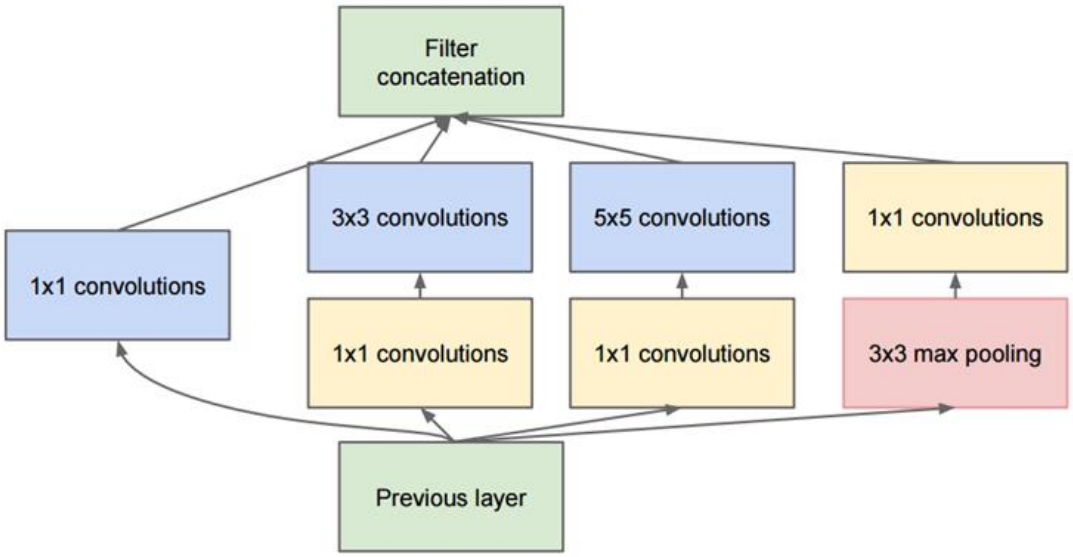
Obrázek 14: Architektura sítě Network in Network (Min, et al., 2013)

Hlavní přínosy této architektury zahrnují:

- Nahrazení lineárního přenosu konvoluční vrstvy dopřednou perceptronovou sítí
- Konvoluce 1x1 napříč mapou příznaků (výsledek základní konvoluce) lépe zachycuje prostorové vztahy napříč příznaky
- Globální průměr závěrečných příznakových map je použit přímo jako vstup pro klasifikační softmax vrstvu (global average pooling).

5.7.3.5. GoogleLeNet a následné modely Inception

Architektura GoogLeNet (Szegedy, et al., 2015) je určena pro rozpoznání obrázku (vítěz ILSVRC 2014 s chybou klasifikace 6.7%). Hlavní výhodou je změna v implementační logice – zatímco předchozí architektury jsou stavěny ryze sekvenčně, architektura GoogLeNet (pozdější verze jsou označovány jako Inception) využívá tzv. Inception moduly (viz. Obrázek 15: Paralelní modul inception, GoogLeNet), tady bloky, které paralelně provádějí zpracování obrazové funkce několika typy konvolučních jader, jejichž výsledek je následně agregován.



Obrázek 15: Paralelní modul inception, GoogLeNet (Szegedy, et al., 2015)

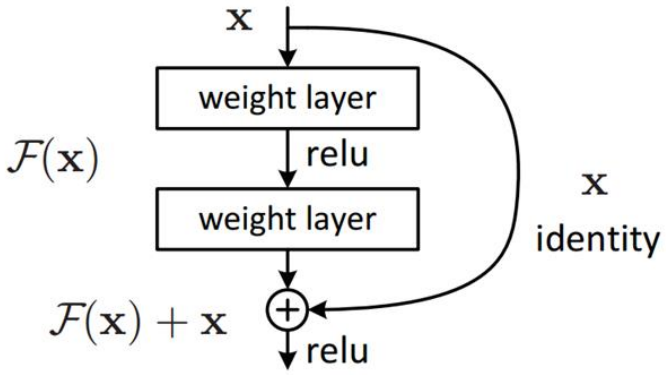
1×1 konvoluce se používají k snížení výpočetních nároků před náročnějšími konvolucemi 3 × 3 a 5 × 5. Tyto vrstvy jsou také následovány ReLU aktivačními vrstvami, zvyšující nelinearitu sítě.

Hlavní výhody zahrnují:

- 9 paralelních inception modulů poskládaných v sérii za sebou
- Nevyužívá plně propojené sítě – před závěrečnou klasifikační vrstvou je v rámci každé příznakové mapy využit sdružování pomocí metody globálního průměru (global average pooling)
- Během testů využívá různí výřezy z obrazu v testovací vrstvy a výsledek je průměrem pravděpodobností z vrstvy softmax
- Novější variace využívají z důvodu úspory parametrů oddělené konvoluce ve vertikálním a horizontálním směru.

5.7.3.6. ResNet

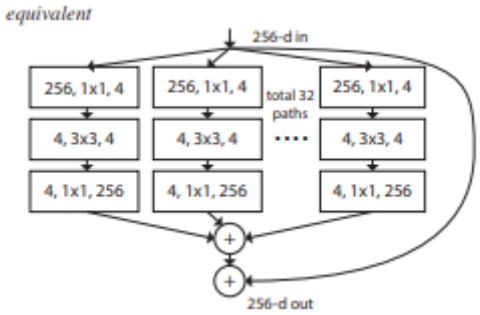
ResNet (He, et al., 2016) je architektura hlubokých neuronových sítí vyvinutá společností Microsoft v divizi Research Asia v roce 2015 využitelná pro rozpoznání a lokalizaci objektů v obrazu (ILSVRC 2015, chybovost 3.6%). Síť ResNet se skládá celkem ze 152 vrstev. Hlavním myšlenkou je tzv. residuální blok (Obrázek 16: Residuální spojení architektury ResNet), jehož hlavním cílem je reprezentace identity. V tradičních reprezentacích sítí lze výstup sítě vyjádřit jako: $H(x) = F(x)$, což lze považovat za novou reprezentaci x která nezachovává reprezentaci původní. Residuální blok propojuje vstup s výstupem, a mezivrstvy se pak učí pouze očekávaný rozdíl. Matematicky pak lze výstup popsat jako: $H(x) = F(x) + x$. Pro dosažení identity (resp. její aproximace) pak postačuje, aby výstupem mezivrstev byl nulový vektor.



Obrázek 16: Residuální spojení architektury ResNet [převzato z (He, et al., 2016)]

Další výhodnou vlastností reziduálního bloku je snadné vyhodnocení zpětného průchodu v algoritmech backpropagation, díky využité operaci sčítání, která gradient distribuuje.

Myšlenka architektury ResNet je dále rozpracována ve variantě ResNeXt (Xie, et al., 2016), která kombinuje residuální spojení s principem inception modulu z architektury Inception (Obrázek 17: Základní kombinace inception modulu a residuálního spojení).

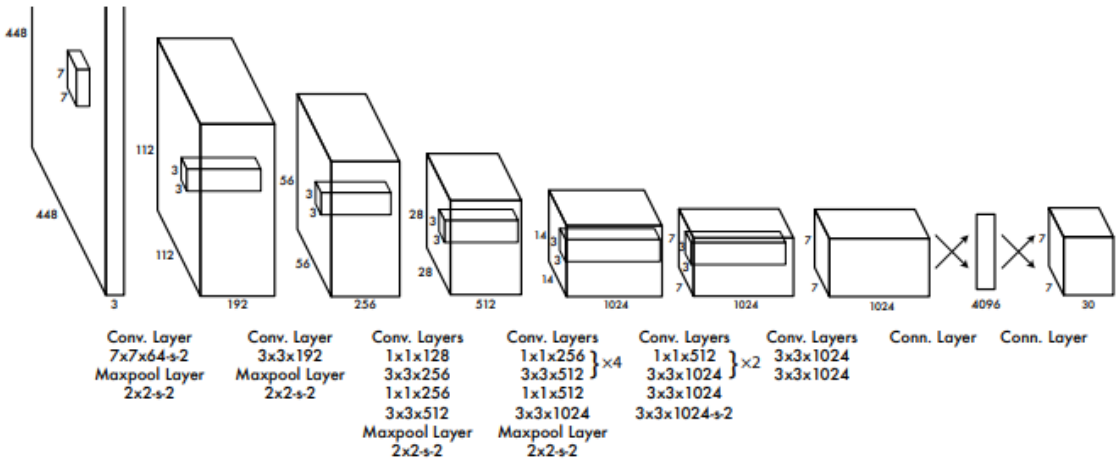


Obrázek 17: Základní kombinace inception modulu a residuálního spojení (Xie, et al., 2016)

Strategie rozdělit-transformovat-sloučit, kterou inception modul využívá se přibližuje reprezentační schopnosti širokých a hustě propojených vrstev, avšak podstatně redukuje výpočetní složitost. Tento modul umožňuje realizace výpočetně efektivní možnosti reprezentace mezivrstvy, a tedy zrychluje učení sítě.

5.7.3.7. YOLO

Architektura YOLO (You Only Look Once) (Redmon, et al., 2016) slouží pro detekci objektů v obraze a určuje jak třídu objektu, tak jeho umístění (ohraničující obdélník). V architektuře YOLO slouží jedna neuronová síť zároveň pro predikci umístění ohraničujících obdélníků i klasifikaci objektu. Architektura využívá celkem 24 konvolučních vrstev (včetně vrstev 1x1) a vychází z principů představených výše v architekturách VGG a NiN.



Obrázek 18: Architektura YOLO - 24 konvolučních vrstev a 2 plně propojené vrstvy (Redmon, et al., 2016)

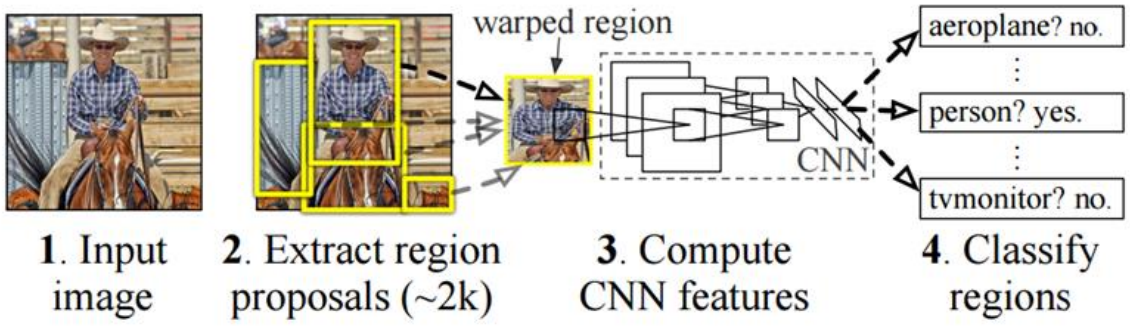
V architektuře YOLO je prochází každý obrázek sítí právě jednou, přičemž výstupem vytváří mřížku o rozměrech *m x m*, kde každá buňka obsahuje informaci o třídě objektu spolu s pravděpodobností. Obdélníky s maximální spolehlivostí jsou označují detekované objekty.

Architektura YOLO je dále rozvíjena v dalších inkarnacích (např. YOLOv2, 2016), které rozšiřují množinu detekovaných objektů a zvyšují přesnost umístění ohraničujících obdélníků.

5.7.3.8. R-CNN, Fast R-CNN a Faster R-CNN

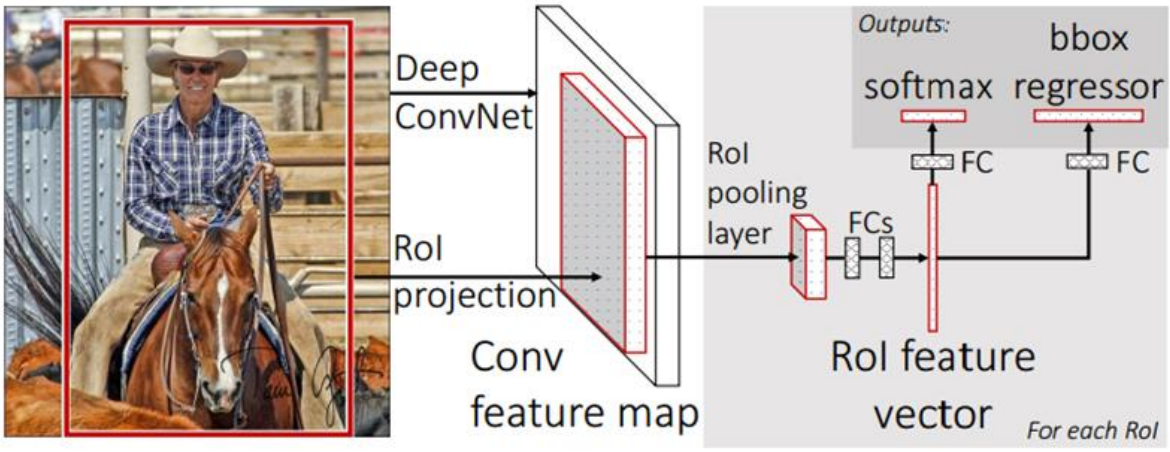
Architektura R-CNN (Girshick, et al., 2016) (a navazující zrychlení Fast R-CNN (Girshick, 2015) a Faster R-CNN (Ren, et al., 2015)) slouží primárně pro detekci objektů v obrazu, resp. označení regionu v němž se obraz nachází.

Fungování R-CNN lze rozdělit do dvou kroků – návrh regionu v němž se objekt nachází a následná klasifikace. R-CNN k tomuto využívá algoritmus „Selective Search“, který generuje 1000-10 000 rozličných oblastí, v nichž se s největší pravděpodobností nějaký objekt nachází. Tyto oblasti jsou následně škálovány (warp deformace) na rozměr, který je vhodný pro učení neuronové sítě (v případě CNN AlexNet), která slouží k výběru příznakového vektoru pro každý region. Tento vektor je použit jako vstup pro lineární SVM, které jsou naučeny pro každou z kategorií a jejichž výsledkem je klasifikace. Zároveň je také použit regresor, který využívá metody *non-maxima suppression* pro výběr nejvhodnějšího z překrývajících se ohraničujících obdélníků.



Obrázek 19: Architektura R-CNN (Girshick, et al., 2016)

Fast R-CNN je založena na potřebě výrazně zrychlit proces učení sítě. Zatím co v základním modelu prochází každý z vybraných regionů (ohraničených oblastí) všemi konvolučními vrstvami, v architektuře FAST R-CNN jsou příznakové mapy pro celý obrázek spočteny pouze jednou a na základě závěrečné příznakové mapy jsou vybírány zájmové oblast (RoI, tedy *Region of Interest*), opět pomocí metody *Selective search*. Tyto oblosti jsou následně klasifikovány softmax vrstvou a vybraný ohraničující obdélník upřesněn. Protože většina výpočtů detekčních sítí je soustředěna v konvolučních vrstvách, dochází k výrazné úspoře výpočetních prostředků.

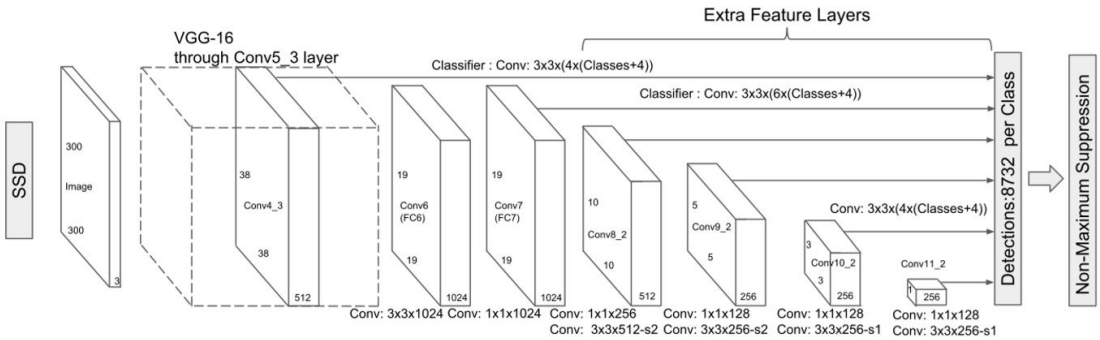


Obrázek 20: Architektura Fast R-CNN (Girshick, 2015)

Architektura Faster R-CNN nahrazuje algoritmus Selective search sloužící pro výběr oblastí regresní neuronovou sítí. Zájmové oblasti jsou vybírány až na základě závěrečné příznakové mapy vzniklé konvolucí. Celý systém tak lze trénovat jako celek a je významně rychlejší.

5.7.3.9. Single Shot Multibox Detector

Architektura Single Shot Multibox Detector (SSMD) vychází z architektury VGG, ze které odstraňuje plně propojené vrstvy a nahrazuje je konvolučními vrstvami. Výstupem těchto vrstev jsou, podobně jako u předchozí architektury YOLO, hraniční obdélníky. Díky použití konvolučních vrstev s různým měřítkem tato architektura lépe detekuje objekty o různé škále.



Obrázek 21: Architektura sítě SSMD

6. Experimentální rozpoznání plevelných rostlin v ranné fázi jejich růstu

Naprostá většina současných výzkumných studií v oblasti rozpoznání plevelných rostlin se zaměřuje na konkrétní aplikační doménu – tedy rozpoznání plevele ve specifické kulturní plodině, a většinou pak pouze odlišení plevele od pěstované plodiny. Tento postup se shoduje se závěry článku „*Design of intelligent decision support systems in agriculture*“ (Hanzlík, et al., 2015), který popisuje tvorbu monotematických znalostních modelů se zaměřením na konkrétní aplikaci. Hlavní výhodou takových systémů je pak obvykle snadnější nalezení vlastností, na jejichž základě lze rozpoznání realizovat. Systém schopný cíleného rozpoznání nejproblematičtějších plevelných rostlin však má výrazně širší aplikační potenciální napříč geografickou oblastí vymezenou výskytem podobných rostlin.

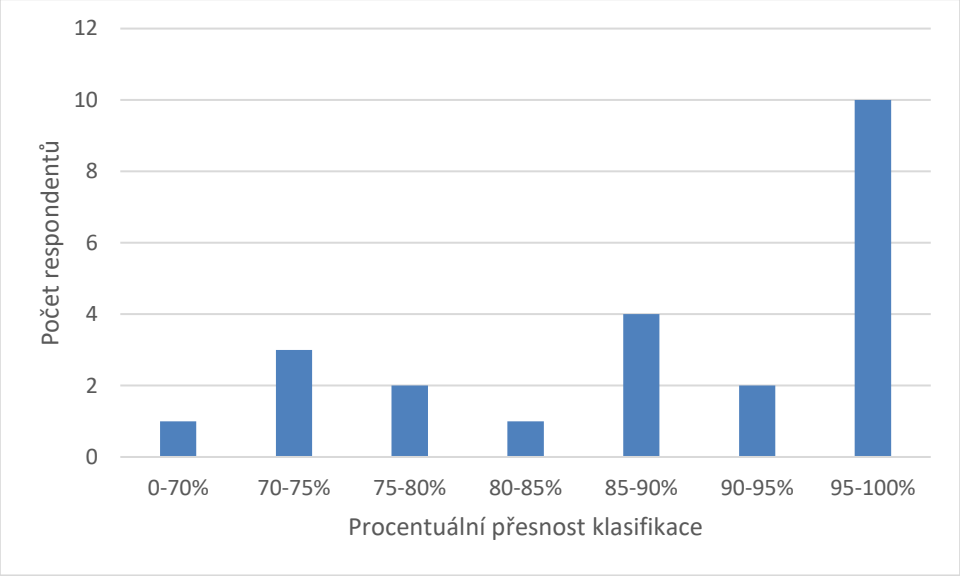
6.1. Specifikace úlohy

Neuronové sítě, jakož i další metody založené na principu učení s učitelem (tedy *supervised learning*) vyžadují pro tvorbu klasifikačního modelu rozsáhlou množinu předem klasifikovaných vzorových příkladů. Některé architektury konvolučních neuronových sítí také vyžadují ruční označení (anotaci) a klasifikaci hledaných objektů ve scéně (např. architektury ze skupiny YOLO). V případě mnoha úloh je tento přístup velmi rozumný a objekty v učební množině dokáže vybrat a označit libovolná osoba (např. rozpoznání dopravních značek, obličejů, SPZ). V případě úlohy klasifikace specifických rostlinných druhů je však nutností spolupráce s expertem, který dokáže druh z obrázku spolehlivě určit. Samotné určení je navíc časově náročné, a ne vždy je výsledná klasifikace dostatečně spolehlivá.

Pro ověření spolehlivosti samotné expertní klasifikace vznikl jednoduchý interaktivní dotazník, jehož cílem bylo ověřit, jaké přesnosti dosahují v klasifikaci plevelných rostlin specialisté ze zemědělské oblasti. Struktura dotazníku byla následující:

1. Dotazy zjišťující profesní charakteristiky respondenta (stupeň a obor vzdělání, vykonávaná profese, délka praxe).
2. Informační část: Přehled rostlin, které jsou předmětem zbytku dotazníku, včetně označení příznaků, na jejichž základě lze rostlinu rozpoznat.
3. Testovací část: určení druhu rostliny (výběr jedné z možností) u 30 náhodně vybraných obrázků ze sady 120 snímků o rozlišení 800x800, přičemž na každém z obrázků je právě jedna označená rostlina.

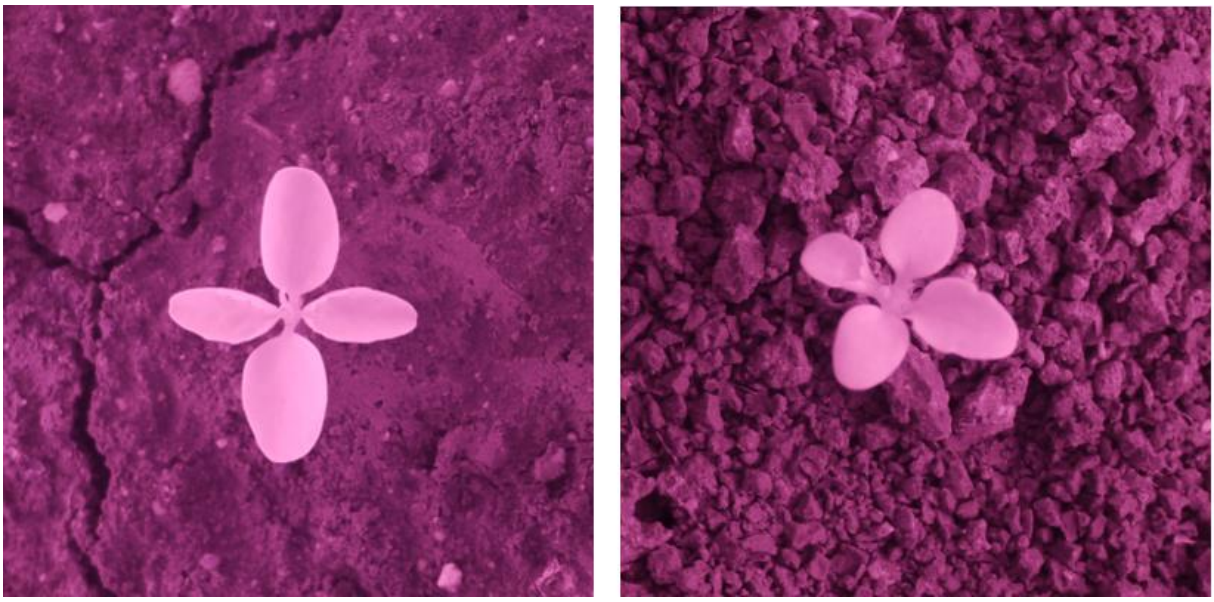
Kompletně dotazník vyplnilo v období 12.3.2016 – 18.2.2017 celkem 120 respondentů. Z těchto respondentů lze na základě vyplněných profesních charakteristik 23 považovat za specialisty v oblasti rostlinné výroby, popř. biologie. Tito respondenti dosáhli průměrné relativní přesnosti klasifikace 89,13%. Detailnější rozdělení do kategorií vytvořených dle intervalů přesnosti klasifikace je uvedeno na Obrázek 22. Mimo skupinu expertů přesnost klasifikace klesá až na 63,54%. Zároveň lze v rámci této skupiny sledovat významnou korelaci mezi časem stráveným vyplňováním dotazníku a přesností klasifikace.



Obrázek 22: Relativní přesnost klasifikace plevelných rostlin specialisty.

Matice záměny vyplývající z expertní klasifikace pak naznačuje, že v některých případech je rozlišení jednotlivých rostlinných druhů bez dalšího kontextu, popřípadě bez možnosti změnit

úhel pohledu skutečně obtížné. Jako příklad uveďme rostliny na Obrázek 23 – v obou případech 4 ze 7 respondentů, kteří obrázek klasifikovali vzájemně zaměnili rostlinný druh.



Obrázek 23: Bažanka roční (vlevo) v přímém srovnání s Violkou rolní (vpravo)

Vzhledem k celkovému počtu relevantních respondentů nelze výsledky považovat za statisticky významné, a výsledek je tedy ryze informativní.

6.2. Vlastní datová množina

Tato dizertační práci se zaměřuje na metody rozpoznání plevele v raných fázích jejich růstu (do čtvrtého týdne od vyklíčení). Tato růstová fáze je specifická rychlými a významnými změnami ve vzhledu rostlin, jakož i značnou podobností jednotlivých rostlinných druhů. Pro reprezentaci této úlohy byla shromážděna množina fotografií[1] 7 druhů dvouděložných plevelných rostlin (*Mercurialis annua, Lamium amplexicaule, Echinochloa crus-galli, Amaranthus retroflexus, Chenopodium album, Fallopia convolvulus, Viola arvensis*) v růstové fázi, kdy má každá

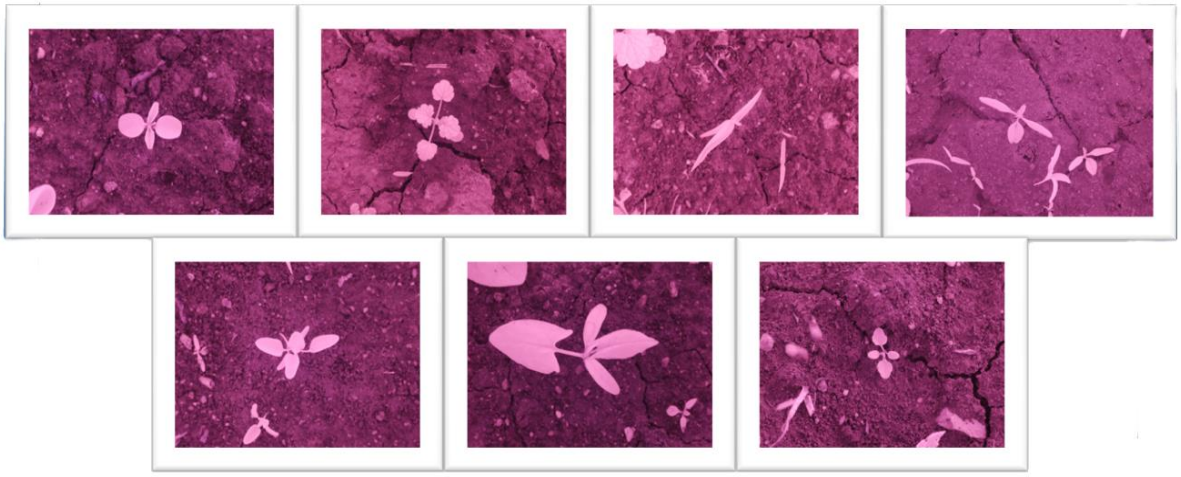
1 Autorem snímků použitých v této práci je Ing. Pavel Hamouz Ph.D., odborný asistent na Fakultě agrobiologie, potravinových a přírodních zdrojů České zemědělské univerzity v Praze

rostlina dva páry děložních lístků (viz. Obrázek 24). Každý rostlinný druh je v této množině zastoupen více než 400 snímky v rozlišení 3,264 x 2,448 pixelů a je uložen ve formátu JPEG.

Tyto rostliny jsou charakteristickými zástupci polních plevelů vyskytujících se v kulturní vegetaci. Jsou zvoleny jednak kvůli svému významnému dopadu na kulturní vegetaci (např. ježatka je často řazena mezi 10 nejškodlivějších plevelných rostlin vůbec) a také kvůli značné podobnosti mezi jednotlivými růžicemi v prvních týdnech růstu (dvojice bažanka – violka a laskavec – merlík jsou si velmi podobné) přičemž ježatka je zvolena jako zástupce. Pro rozpoznání je tedy nutné využít širokou škálu specifických příznaků.

Snímání bylo s ohledem na algoritmické zpracování pro aplikace strojového učení provedeno ručně ze stabilní vzdálenosti tak, že výsledný obraz obsahuje plevelnou rostlinu co nejblíže svému středu. Pro usnadnění rozlišení rostlin od pozadí byla při snímání využita metoda vycházející z principu multispektrálního snímkování, která za pomoci filtru umístěného za objektiv mění interval snímané vlnové délky záření směrem blízko-infračervenému záření (NIR). Vzhledem k možnému zvýšení variability úlohy související s osvětlením jsou v momentě snímání předpokládány stabilizované podmínky osvětlení. V každém obraze se může vyskytovat vícero rostlin jak jednoho, tak i více druhů, i další objekty přírodního původu (kameny, suchá stébla, další rostliny aj.).

Každý obraz byl následně expertně klasifikován na základě druhové příslušnosti centrálního objektu. Výsledkem tohoto kroku tedy bude reprezentativní množina obrazů vybraných plevelných rostlin. Zároveň pak byla pro každý rostlinný druh z této množiny zaznamenána charakteristická množina vizuálních vlastností, které k rozpoznání používá expert. Tuto znalost lze teoreticky využít pro návrh algoritmů vhodných pro rozlišení jednotlivých druhů rostlin.

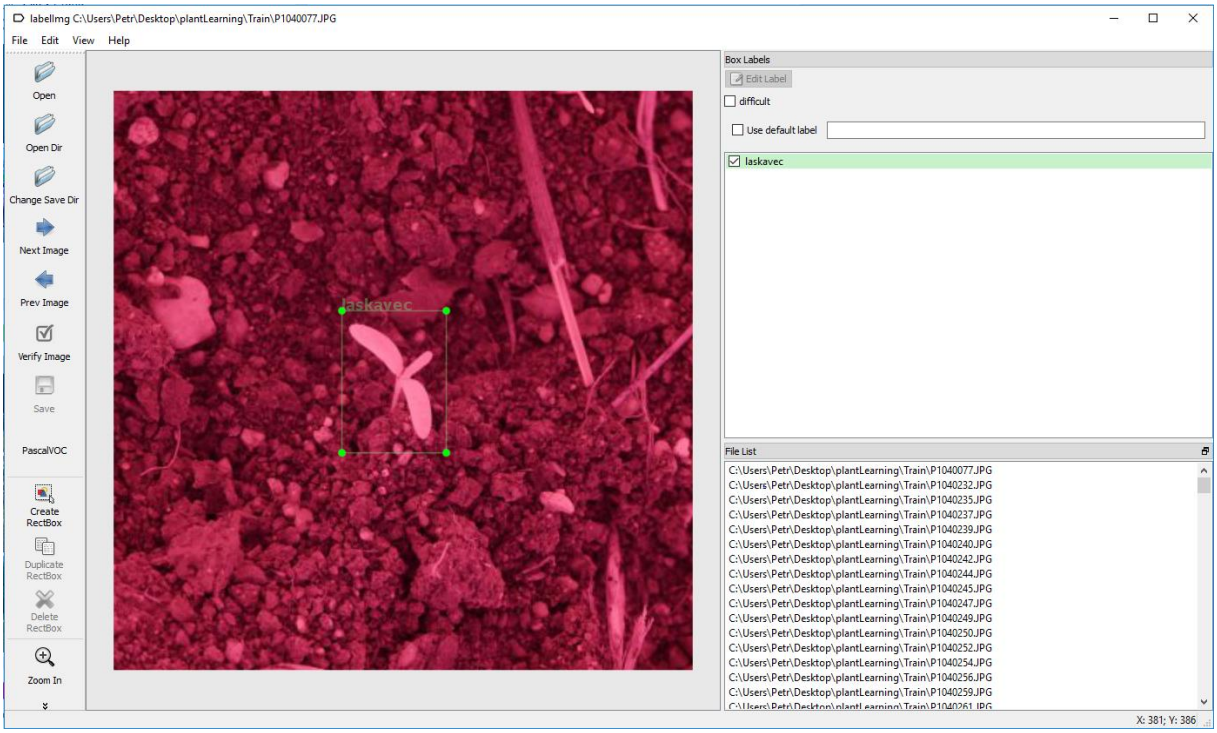


Obrázek 24: Ukázka vybraných plevelných rostlin na snímcích, na nichž je rozdíl mezi jednotlivými druhy nejpatrnější [autor snímků: Ing. Pavel Hamouz, Ph.D., FAPPZ ČZU]

Pro ověření použitelnosti klasifikačních metod v reálných podmínkách, v níž se rostliny vzájemně překrývají a které jsou tedy bližší aplikaci na polních mechanizaci, byla shromážděna verifikační množina celkem 200 obrázků stejných druhů rostlin, která vznikla z výřezů množiny původní. V rámci této množiny se jednotlivé rostliny vzájemně překrývají, v mnoha případech také obsahuje zástupce vícero druhů rostlin.

Veškeré shromážděné obrázky byly následně ručně anotovány v programu LabelImage[2].

2 https://github.com/tzutalin/labelImg



Obrázek 25: Ukázka ruční anotace rostliny v programu LabelImage[2]

6.3. Společné metody předzpracování a segmentace obrazu

Základem většiny metod rozpoznání obrazu založených na extrakci příznaků je předzpracování, jehož cílem je reprezentovat obraz způsobem, který omezuje vliv šumu, ale zároveň zachovává cílový objekt nepoškozený. Vzhledem k metodě snímání jejímž cílem je zvýraznění rostliny, je v rámci dostupné obrazové množiny segmentace poměrně přímočará a je možné použít základní metody bez větší ztráty informace.

Použité metody společného předzpracování pak zahrnuji:

- Převod do tónů šedi na základě metody rozkladu světla (luminosity method), dle rovnice I = 0.2126 * R + 0.7152 * G + 0.0722 * B.
- Redukce šumu pomocí mediánového filtru.
- Segmentace obrazu pomocí jednoprůchodového algoritmu globálního prahování, využívajícím výpočet prahového hodnoty pomocí minimální křížové entropii (Li & Tam, 1998).

Tato procedura je vizuálně shrnuta v Obrázek 26.

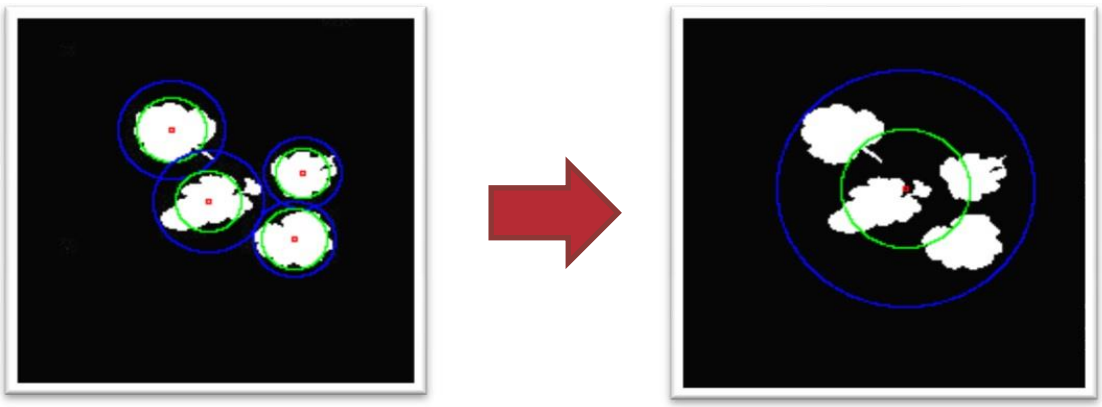
Obrázek 26: Předzpracování obrazu, segmentace

Ve výsledném binárním obrazu jsou detekovány souvislé oblasti, které přesahují minimální stanovenou velikost pomocí algoritmu využívající strategii depth-first a Moorovo okolí pro určení spojitých oblastí. V rámci shromážděné učební sady se předpokládá, že objekt, který má být rozpoznán je umístěn nejblíže přirozeného středu obrazu. Tento objekt je po segmentaci vybrán jako kandidát pro další zpracování. Vzhledem k poměrně jednoduché metodě segmentace však někdy, vzhledem k nevýrazné (přistíněné) části listové stopky, dochází

k rozpadu objektu do několika částí. Pro řešení jsme navrhl a využíval následující heuristické řešení:

- Každá nálezná oblast *O* je popsána pomocí svého těžiště, maximální vzdálenosti mezi těžištěm a hranou objektu (d) a průměrem kruhu s plochou stejně velkou jako je plocha nalezené oblasti (ed).
- Výpočet poměru vzdáleností (r); r = d/ed.
- Je-li r> 0.7 (široký list) vzdálenost (d) je použita jako kritická vzdálenost (θ), v opačném případě je použit poloměr kruhu o stejném obsahu (ed).
- Je-li Euklidovská vzdálenost mezi těžišti dvou oblastí menší než součet jejich kritických vzdáleností θ, budou oblasti sloučeny.

Výsledek tohoto heuristického řešení je naznačen v Obrázek 27. Po výběru cílového rostliny jsou zbylé binární objekty ze scény odstraněny.

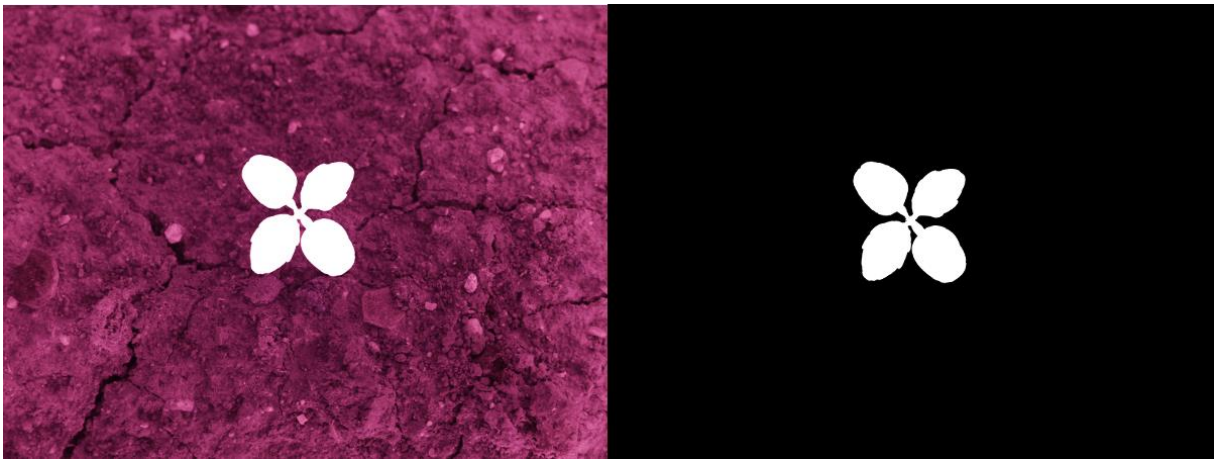


Obrázek 27: Sdružení spojitých oblastí v obrazu na základě kritické vzdálenosti (modře maximální vzdálenost od těžiště, zeleně obvod kruhu o shodném poloměru)

Cílem tohoto předzpracování není jemnější segmentace, pouze rozlišení popředí (rostliny) a pozadí. Pro vyhodnocení úspěšnosti použití tohoto snadného postupu bylo ručně segmentováno 100 náhodných výřezů obsahujících jednu rostlinu. Tyto ruční výřezy byly srovnány s výstupem algoritmického řešení (viz. Obrázek 28), přičemž pro srovnání byla využita Precision/Recall křivka a odpovídající statistika F-skóre vyjádřená jako:

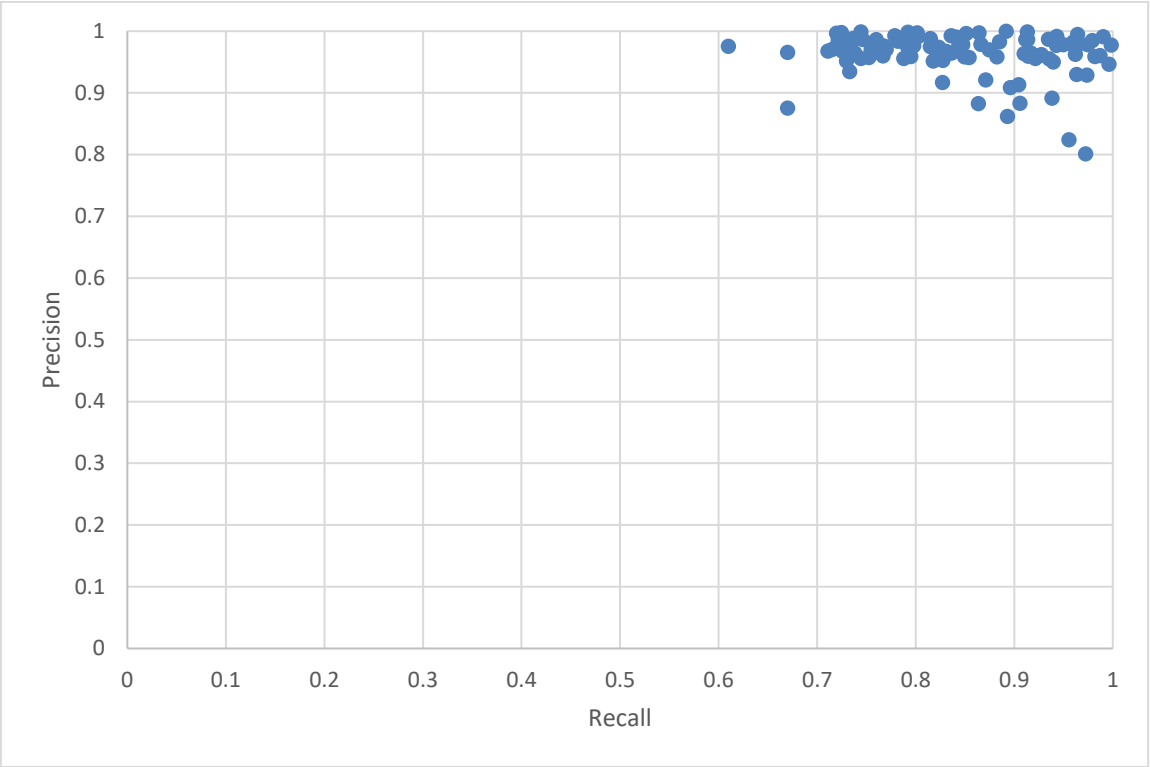
$$F = \frac{2\mathrm{PR}}{(\mathrm{P} + \mathrm{R})},$$

kde P je míra přesnosti vyjádřená jako poměr správně označených pixelů popředí (TP) ku celkovému součtu pixelů označených algoritmem jako popředí (tedy P = TP / (TP + FP)) a R je poměr správně označených pixelů popředí (TP) ku součtu těchto pixelů s body objektu, které nebyly označeny jako popředí (teda R = TP / (TP + FN)).



Obrázek 28: Ukázka porovnání manuální a automatické segmentace rostliny (Bažanka)

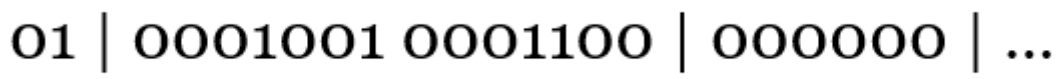
Při použití tohoto jednoduchého kritéria je průměrné F-skóre 0,91, což lze považovat za přijatelnou ztrátu informace. Z obrázku 29 je pak patrné, že počet chybně nalezených pixelů (míra precision) je celkově poměrně nízký, a větším problémem je tak ztráta informace způsobena nenalezením některých pixelů. Vzhledem k celkovému skóre lze předpokládat, že, že metoda segmentace nebude mít významnější vliv na celkovou kvalitu extrakce příznaků z takto vzniklých objektů.



Obrázek 29: Graf Precision-Recall zachycující kvalitu segmentace u 100 vybraných obrázků ze shromážděné množiny.

6.4. Společná metoda klasifikace příznakových vektorů

Rostliny se obecně vyznačují značnou variabilitou tvarů v rámci jednoho druhu. Proto se pro klasifikaci jako nejvhodnější jeví metody umělé inteligence schopné generalizace. V tomto případě byly pro klasifikaci na základě příznakových vektorů využity dopředné neuronové sítě s jednou skrytou vrstvou, které využívají pro učení algoritmus improved resilient backpropagation (Igel & Hüsken, 2000). Velikost vstupní vrstvy je dána vstupním vektor, velikost výstupní vrstvy odpovídá počtu kategorií. Velikost a počet (pouze 1 nebo 2) skrytých vrstev je vybírána experimentálně pomocí Lamarkova genetického algoritmu na základě fitness funkce (přesnosti klasifikace) v omezeném počtu iterací (10.000).



Obrázek 30: Kodifikace architektury použitelná pro genetický algoritmus – první 2 bity udávají počet vrstev, druhá oddělená část počet neuronů v jednotlivých vrstvách, třetí míru učení v intervalu 0.01 – 0.67

Pro potřeby učení sítě je vždy množina rozdělena do tří částí – učební sada (80%), validační sada (10%), a testovací sada (10%). Algoritmus učení využívá základní strategii zabraňující přeučení sítě, který porovnává chybu klasifikace na učební množině s množinou validační. Výsledná přesnost naučené sítě je na závěr vyhodnocena na naučené síti. Vzhledem k omezené velikosti učební sady ve vztahu se složitostí úlohy a možném vlivu testovací množiny na výsledek je každý klasifikační experiment opakován 10x vždy s náhodnou kombinací učební a testovací sady. Konečný výsledek je poté vyjádřen jako aritmetický průměr výsledků všech provedených iterací.

Jako kritérium hodnocení je primárně používána procentuální přesnost klasifikace, jako pomocná kritéria jsou opět použity míry precision (P) a recall (R) vycházející z matice záměny, které jsou kombinovány do podoby F1-skóre. Pro každou třídu *i* pak platí, že:

$$(46) \quad P_i = \frac{TP_i}{(TP_i+FP_i)} \ , \ R_i = \frac{TP_i}{(TP_i+FN_i)} \ , \ F1_i = \frac{2\mathrm{P}_i\mathrm{R}_i}{(\mathrm{P}_i+\mathrm{R}_i)}$$

Kde TP je absolutní počet správně určených instancí třídy *i*, FP je počet instancí nesprávně označených jako třída *i*, a FN je počet instancí, které nebyly označeny jako instance třídy *i* ačkoliv do ní patří. Práce využívá pouze nejpravděpodobnější variantu klasifikace.

Dopředné vícevrstvé neuronové sítě byly implementovány ve frameworku Encog 3.0[3] (Heaton, 2015) pro programovací jazyk JAVA.

[3] Encog Machine Learning Framework, https://github.com/encog/encog-java-core

6.5. Metody rozšíření datové základny

Klasifikační metody založené na principu učení s učitelem (tedy *supervised learning*) vyžadují pro tvorbu modelu rozsáhlou, a především reprezentativní množinu předem klasifikovaných vzorových příkladů. V oblasti rozpoznání rostlin je běžná velká variabilita vzhledu v rámci jednoho druhu, přičemž vliv na celkový vzhled mohou mít také proměnné, jako je prostředí (sucho, vlhko, stín, typ půdy aj.), nebo mechanické či chemické poškození. Jak naznačuje předchozí kapitola, shromáždění takové množiny může být nesnadné.

Z toho důvodu jsou často využívány různé transformace obrazu, které zvyšují variabilitu učební množiny. Obecně lze taková rozšíření popsat jako afinní transformace dle rovnic:

$$x' = a_0 + a_1 x + a_2 y \tag{44}$$

$$y' = b_0 + b_1 x + b_2 y \tag{45}$$

Mezi tyto základní transformace lze zařadit:

- Transformace překlopením (odrazem) horizontální, vertikální a kombinovaná
- Zvětšení scény s faktorem $\omega > 0$ a následným výřezem na původní velikost
- Rotace scény v postupných krocích o úhel $\alpha = 15°$
- Lineární warp transformace obrazu, přičemž projekční mřížka je daná posunem rohů obrazové scény o v náhodném intervalu daného 1-10% výšky a šířky obrazové scény

Tyto metody jsou v dalším průběhu řešení využívány tam, kde je to relevantní, tedy v případě metod extrakce příznaků, které jsou vůči některé z těchto degradací obrazové funkce citlivé.

6.6. Přístupy založené na extrakci příznaků

Pro popis objektů v obrazu bylo využito několik rozličných metod extrakce příznaků vycházejících z teoretických poznatků v kapitole 5.5. Vybrané metody, které byly úspěšně využity pro rozpoznání rostlin (např. listů stromů) v literatuře často nebyly experimentálně ověřeny v oblasti rozpoznání plevelných rostlin, nebo byly využity za ryze ideálních podmínek. Tato kapitola shrnuje principy a modifikace použitých metod extrakce příznaků, které jsou později využity pro rozpoznání.

6.6.1. Hand-crafted features

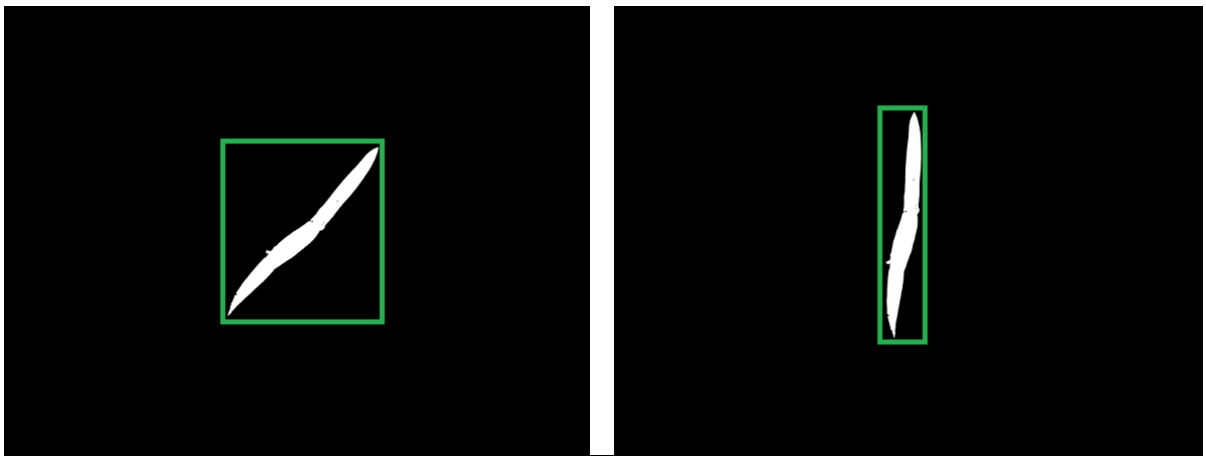
Hand-crafted features jsou příznaky navržené tak, aby odpovídali jednoduchému popisu, který pro tvar použije člověk. Existuje celá řada těchto příznaků, z nichž některé jsou shrnuty v kapitole 5.5.1. Pro rozpoznání objektů v obraze byly v rámci práce využity příznaky z kapitoly 5.5.1, které jsou alespoň částečně citlivé vůči změnám velikosti objektu:

	Deskriptor	**Výpočet**	**Popis.**
1	Kompaktnost	a/p^2	Poměr plochy objektu (a) a kvadrátu jeho obvodu (p)
2	Jednolitost	a/c	Poměr plochy objektu (a) a jeho konvexního obalu (c)
3	Konvexnost	p/r	Poměr obvodu objektu (p) a obvodu jeho konvexního obalu (r)
5	Kulatost	$\frac{4\pi a}{p^2}$	Míra podobnosti mezi tvarem listu a kruhem
6	Poměr stran	h/w	Poměr délky a šířky ohraničujícího obdélníku
8	Hranatost	$\frac{h * w}{a}$	Podobnost mezi plochou tvaru (a) a obdélníkem daným přirozenou výškou (h) a šířkou tvaru (w)
9	-	d/h	Poměr mezi maximální vzdáleností mezi dvěma body na

			hraně objektu (d) a jeho přirozenou délkou (h)
10	-	p/d	Poměr obvodu (p) a maximální vzdáleností mezi dvěma body na hraně objektu (d)

Tabulka 2: Použité deskriptory hand-crafted features

Míry využívající ohraničující obdélník, které jsou vhodné zejména na určení podlouhlosti objektu jsou závislé na rotaci objektu (viz. Obrázek 31). Směr objektu v obraze je proto normalizován rotací objektu v obrazu tak, že hlavní komponenta je umístěna rovnoběžná s osou y.



Obrázek 31: Vliv rotace na tvar ohraničujícího obdélníku

Pro trénink sítě je pak využito 10 výše uvedených poměrových veličin. Při využití plného rozlišení obrázku se přesnost klasifikace pohybuje kolem 59,6%, přičemž matice záměny v tomto případě naznačuje, že na základě těchto příznaků lze spolehlivě odlišit pouze ježatku kuří nohu.

		Kalasifikace neuronovou sítí									
		Bazanka Roční	Hluchavka Objímavá	Ježatka Kuří Noha	Laskavec Ohnutý	Merlík Bílý	Opletka Obecná	Violka Rolní	R	P	F-stats
Skutečná třída	Bazanka Roční	37	3	0	0	0	0	0	0.93	0.62	0.74
	Hluchavka Objímavá	6	25	0	0	6	0	3	0.63	0.41	0.50
	Ježatka Kuří Noha	0	0	37	3	0	0	0	0.93	1.00	0.96
	Laskavec Ohnutý	0	12	0	12	16	0	0	0.30	0.63	0.41
	Merlík Bílý	0	4	0	4	16	16	0	0.40	0.41	0.41
	Opletka Obecná	5	1	0	0	1	33	0	0.83	0.66	0.73
	Violka Rolní	12	16	0	0	0	1	7	0.19	0.70	0.30
								Celkem	0.60	0.63	0.58

Obrázek 32: Matice záměny, hand-crafted features

6.6.2. Huovy invariantní momenty

Huovy invariantní momenty (Hu, 1962) jsou množina deskriptorů objektu, které jsou invariantní vůči translaci a v normalizované podobě také vůči změnám ve škále a rotacím. Kadir et al. (2011) dosáhli úspěšnosti klasifikace listů stromů až 72,40 %. Byla však použita vlastní, nikoliv standardní množina objektů. Pro ověření použitelnosti pro řešení úlohy rozpoznání plevelných rostlin bylo použito 7 Huových momentů, které tvoří výsledný vektor použitý pro učení neuronové sítě. Vzhledem k výše zmíněné invarianci vůči základním degradacím obrazu jsou v rozšířené učební množině použity pouze deformace vedoucí ke změně úhlu projekce a k „warp“ roztažením objektu.

Řešení pomocí Huových invariantů dosahuje na testovací množině při použití obrazu v plném rozlišení poměrně nízké přesnosti klasifikace 40%. Veskrze lze pak konstatovat, že výsledky tohoto extraktoru jsou pro rozpoznání rostliny nedostatečné, jak je patrné z matice záměny. Zatím co přístupy založené na

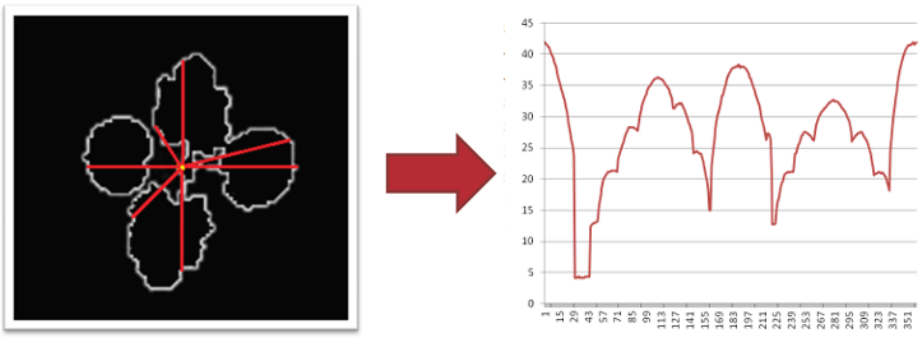
		Kalasifikace neuronovou sítí									
		Bazanka Roční	Hluchavka Objímavá	Ježatka Kuří Noha	Laskavec Ohnutý	Merlík Bílý	Opletka Obecná	Violka Rolní	R	P	F-stats
Skutečná třída	Bazanka Roční	34	5	0	0	0	1	0	0.85	0.40	0.54
	Hluchavka Objímavá	10	19	0	0	2	7	2	0.48	0.25	0.32
	Ježatka Kuří Noha	3	9	12	6	0	1	9	0.30	0.71	0.42
	Laskavec Ohnutý	8	11	2	11	5	1	2	0.28	0.46	0.34
	Merlík Bílý	13	17	2	0	6	2	0	0.15	0.38	0.21
	Opletka Obecná	13	5	0	2	1	16	3	0.40	0.53	0.46
	Violka Rolní	5	11	1	5	2	2	14	0.35	0.47	0.40
								Celkem	0.40	0.45	0.39

Obrázek 33: Matice záměny, Huovy momenty

6.6.3. Rozšířený deskriptor CCD

První využitý deskriptor je založen na popisu tvarových vlastností rostliny v obraze pomocí množiny vzdáleností mezi těžištěm a hranou objektu. Vstupním objektem pro tento algoritmus je binární objekt v obraze vzniklý metodou předzpracování popsanou v kapitole 6.3. Pro tento objekt je následně spočtena hlavní osa metodou analýzy hlavních komponent, která je použita jako výchozí průmět *p* použitý pro měření vzdáleností mezi těžištěm a hranou. Tato vede k významnému snížení vlivu rotace na klasifikaci, neboť je nalezena stabilní poloha.

Pro objekt je pak nalezena množina polopřímek s počátečním bodem v těžišti, kdy každá další polopřímka svírá s předcházející úhel ρ. Celkový počet měření je tak dán vzorcem $2\pi/\rho$. Pro každou z těchto polopřímek je pak vypočtena Euklidovská vzdálenost mezi hranou a těžištěm. V případě tohoto experimentu je využita úhel $\rho = 1°$.



Obrázek 34: Převod obrázku na funkci maximálních vzdáleností od těžiště

Vzhledem k časté tvarové složitosti listových růžic plevelných rostlin je použita kombinace dvou vzdáleností – maximální vzdálenosti, která popisuje vnější hranu objektu (viz. Obrázek 34) a druhé nejdelší vzdálenosti k hraně. V případě, že je objekt kompaktní (neobsahuje mezery), tedy ve směru polopřímky existuje pouze jeden průsečík s hranu, je jako druhý bod použita znovu maximální vzdálenost.

Tento morfologický popis je invariantní vůči změnám pozice. Žádoucí invariance vůči rotací je dosaženo tak, že jako první bod každé funkce je použit průsečík hrany se sečnou odpovídající směrem hlavní osou, spolu s použití rotovaných poloh objektu v učební množině Invariance vůči škálování je dosaženo normalizací výsledných funkcí na vektor o velikosti <0;1>.

Výstupem je pak vektor vzdáleností, v němž jsou jednotlivé dvojice maximální a druhé největší vzdálenosti poskládány za sebou dle pořadí získaných polopřímek. Výsledný vektor použitý pro klasifikaci se tedy skládá z 720 hodnot.

Výsledná matice záměny sloužící pro klasifikaci rostlin dle tvaru je uvedena v obrázku 35. je zjevné, že tvarová podobnost některých rostlinných druhů je značná a použitý deskriptor CCD není dostatečně přesný pro rozlišení jednotlivých rostlinných druhů.

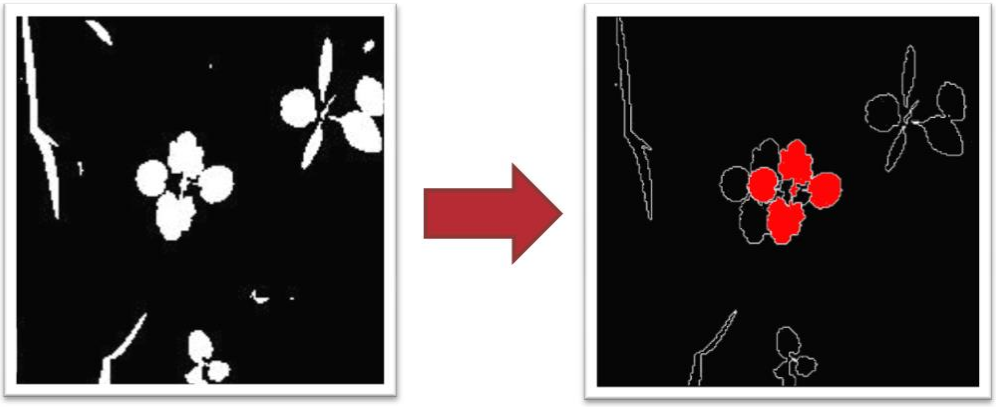
		Kalasifikace neuronovou sítí									
		Bazanka Roční	Hluchavka Objímavá	Ježatka Kuří Noha	Laskavec Ohnutý	Merlík Bílý	Opletka Obecná	Violka Rolní	R	P	F-stats
Skutečná třída	Bazanka Roční	32	0	0	0	0	8	0	0.80	0.80	0.80
	Hluchavka Objímavá	4	26	0	0	2	0	8	0.65	1.00	0.79
	Ježatka Kuří Noha	0	0	40	0	0	0	0	1.00	1.00	1.00
	Laskavec Ohnutý	0	0	0	34	6	0	0	0.85	0.56	0.67
	Merlík Bílý	4	0	0	17	8	3	8	0.20	0.31	0.24
	Opletka Obecná	0	0	0	0	0	40	0	1.00	0.78	0.88
	Violka Rolní	0	0	0	10	10	0	20	0.50	0.56	0.53
								Celkem	0.71	0.71	0.70

Obrázek 35: Matice záměny, rozšířené CCD

6.6.4. Přímá klasifikace neuronovou sítí

Přímá klasifikace objektu jednoduchou dopřednou síť je v případě nalezení normalizované reprezentace objektu částečně možná. Podobný přístup také volí (Yang, et al., 2002) pro rozlišení plevelných rostlin a kukuřice. Tento přístup však vyžaduje rozsáhlou a lze říct, že úplnou učební množinu.

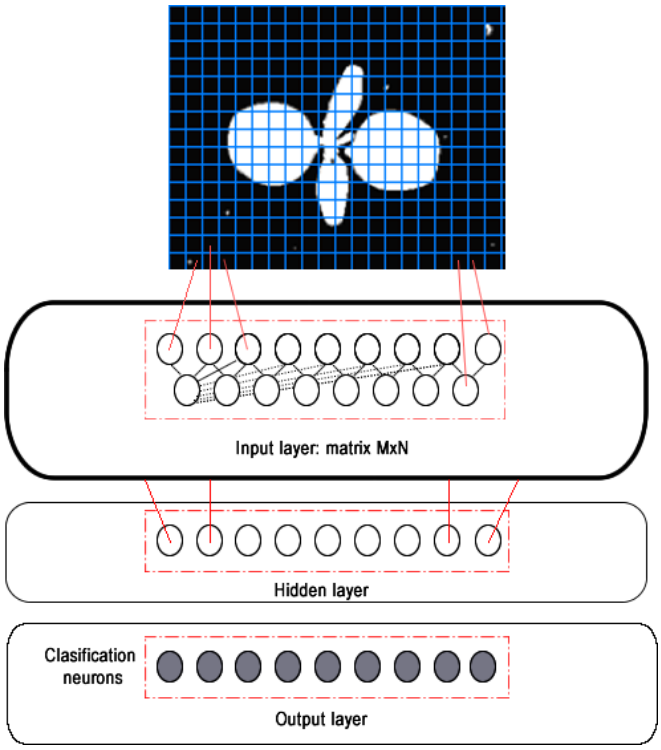
Pro nalezení normalizované reprezentace objektu ve scéně je nejprve využit algoritmus pro detekci spojitých binárních komponent. Objekt nejblíže středu obrazu je následně vybrán jako kandidát a posunut do skutečného středu obrazu na základě vektoru posunu daném vzdáleností mezi těžištěm binárního objektu a středem obrázku. Tato transformace je naznačena v Obrázek 36.



Obrázek 36: Nalezení rostliny a její posun do středu obrazu

Objekt je následně natočen podle hlavní osy do pozice, v níž je hlavní komponent binárního objektu rovnoběžný s osou *y*. Následně je pak zmenšen/zvětšen pomocí bikubické interpolace tak, aby zabíral tři čtvrtiny scény. Množina takto vzniklých obrázků je následně rozšířena pomocí technik pro augmentaci datové množiny v kapitole Specifikace úlohy6.1, včetně rotace a warp transformace.

Pro učení neuronové sítě je pak využit tedy použit očištěný segmentovaný binární objekt normalizované velikosti a pozice. Celý obrázek je následně přímo převeden na vstupní vektor neuronové sítě (viz. Obrázek 37: Klasifikace obrázku pomocí dopředné neuronové sítě). Velikost vstupního vektoru je tak dána rozměry vstupní obrazové funkce, vnitřní architektura je zvolena experimentálně pomocí genetického algoritmu.



Obrázek 37: Klasifikace obrázku pomocí dopředné neuronové sítě

V případě přímé klasifikace neuronovou sítí byl vzhledem k výpočetním omezením použity pouze obrázky o rozměrech 400x400 pixelů. Výsledná procentuální přesnost klasifikace dosahuje 29%, v případě využití obrázků o rozměrech 200x200 pak klesá až na 28%. Ačkoliv je tato přesnost vyšší než při využití náhodného odhadu v porovnání s metodami předchozími ji lze považovat za ryze nedostatečnou. Lze také usuzovat, že učební množina nedostatečně pokrývá rozmanitost problémové oblasti.

6.6.5. Local Binary Patterns

Pro rozpoznání samotných objektů byly využity také binární atributy LBP (viz. kapitola 5.5.7). Výhodou tohoto příznaku je nezávislost vůči změnám měřítka a posunům v jasu a kontrastu obrazu a pozici. Tyto vlastnosti jsou užitečné zejména v reálných podmínkách v exteriérech, kdy dochází ke změnám v osvětlení a zastíněné.

Vzhledem k tomu, že vzory na pozadí jsou v tuhle chvíli pro rozpoznání rostlin irelevantní, jsou vzory extrahovány pouze v oblastech daných binární maskou objektu, která je získána postupem uvedeným v kapitole 6.3. Pro extrakci LBP atributů je pak použita černo-bílá reprezentace obrazu, vzniklá metodou rozkladu světla.

Operátor LBP lze interpretovat dvěma základními způsoby – jako sadu bitů, či jako kategorickou proměnnou z intervalu $<0, 256)$ z těchto bitů odvozenou. Spojení s neuronovou sítí lze realizovat přímo, je však vysoce datově neefektivní, neboť pro každý bod použitý jako střed okolí vzniká řetězec P binárních hodnot.

Možných řešení je několik:

- Převod operátoru na desetinné číslo, kde každá hodnota odpovídá jednomu desetinnému místu.
- Využití normalizovaného histogramu kategorií.
- Výpočet počtu změn z 0 na 1 a naopak (p) a určení počtu nulových bitů (q) pro každý operátor.

V práci je využíván Weberův operátor LBP o poloměru 8 pixelů, který zaznamenává intenzity osmi rovnoměrně rozdělených bodů ve svém okolí. Na základě normalizovaného histogramu relativních četnosti binárních vzorů v objektu lze dosáhnout relativní přesnosti klasifikace 71%. Matice záměny pro tuto metodu pak ukazuje, že pomocí LBP lze poměrně úspěšně rozpoznat hluchavku, která se vyznačuje velmi výraznou žilnatinou listů a z větší části také merlík bílý.

		Kalasifikace neuronovou sítí									
		Bazanka Roční	Hluchavka Objímavá	Ježatka Kuří Noha	Laskavec Ohnutý	Merlík Bílý	Opletka Obecná	Violka Rolní	R	P	F-stats
Skutečná třída	Bazanka Roční	27	0	2	5	4	0	2	0.71	0.60	0.65
	Hluchavka Objímavá	0	39	0	1	0	0	0	0.98	0.98	0.98
	Ježatka Kuří Noha	4	0	17	2	15	1	1	0.43	0.65	0.52
	Laskavec Ohnutý	3	0	0	26	5	3	3	0.65	0.63	0.64
	Merlík Bílý	2	1	4	0	30	0	3	0.75	0.53	0.62
	Opletka Obecná	1	0	2	2	3	32	0	0.80	0.84	0.82
	Violka Rolní	8	0	1	5	0	2	24	0.60	0.73	0.66
								Celkem	0.70	0.71	0.70

Obrázek 38: Matice záměny, LBP

6.7. Konvoluční neuronové sítě a přímá klasifikace obrazu

Krom standardních metod založených na extrakci příznakových vektorů z obrazu a jejich následné klasifikaci dopřednou neuronovou sítí byly pro rozpoznání rostlin použity také konvoluční neuronové sítě, tedy metody, které nevyžadují cílené předzpracování obrazu. Tyto sítě v rámci své struktury (zejména konvolučních a sdružovacích vrstev) extrahují významně větší množství příznaků a sdružují je do tzv. příznakových map. V rámci práce byly použity modely ResNet, Xception a Yolo. Pro implementaci všech těchto konvolučních neuronových byly využity prvky frameworku Deeplearning4J [4].

[4] Deeplearning4j: Open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0. http://deeplearning4j.org

6.7.1. Architektura ResNet

Architektura ResNet je variantou velmi hluboké neuronové sítě, která v architektuře zavádí residuální blok. Díky residuálnímu toku se síť v každé konvoluční vrstvě zaměřuje na reprezentaci identity.

Učení sítě vyžaduje množinu předem připravených obrazů, rozdělených do jednotlivých kategorii. Množina je rozšířena veškerými technikami augmentace datové základny z kapitoly 6.5. Architektura ResNet50 (popis architektury je vzhledem k rozsahu uveden v příloze 1) byla nově implementována za použití knihoven, které jsou součástí frameworku Deeplearning4J. Síť byla učena po dobu 2500 iterací, s podmínkou časného ukončení, začne-li stoupat chyba na validační sadě, přičemž pro validační a testovací sadu je použito 10% použitých obrázků. Pro aktualizaci hodnot vah sítě je použita metoda RMSProp s rychlostí učení (learning rate) 0.0001 a momentem 0.1.

Základní architektura ResNet50 využívá obrázku o rozlišení 224x224 pixelů, přičemž relativní přesnost klasifikace dosahuje 89%. Matice záměny v tomto případě potvrzuje zejména značnou podobnost merlíku a laskavce v časných fázích růstu:

		Kalasifikace neuronovou sítí									
		Bazanka Roční	Hluchavka Objímavá	Ježatka Kuří Noha	Laskavec Ohnutý	Merlík Bílý	Opletka Obecná	Violka Rolní	R	P	F-stats
Skutečná třída	Bazanka Roční	35	0	0	1	2	0	2	0.92	0.95	0.93
	Hluchavka Objímavá	0	39	0	1	0	0	0	0.98	0.95	0.96
	Ježatka Kuří Noha	0	0	39	1	0		0	0.98	0.95	0.96
	Laskavec Ohnutý	0	0	0	33	6	1	0	0.83	0.79	0.80
	Merlík Bílý	0	1	2	4	31	0	2	0.78	0.79	0.78
	Opletka Obecná	0	0	0	0	0	40	0	1.00	0.93	0.96
	Violka Rolní	2	1	0	2	0	2	33	0.83	0.89	0.86
								Celkem	0.90	0.89	0.90

Obrázek 39: Matice záměny, síť ResNet

6.7.2. Architektura Xception

Architektura Xception je moderní variantou neuronové sít, která kombinuje poznatky z architektury GoogLeNet (paralelní inception bloky) a residuální blok architektury ResNet. Díky této vlastnosti se vyznačuje vysokou přesností klasifikace a přijatelnou rychlostí učení.

Učení sítě vyžaduje množinu předem připravených obrazů, rozdělených do jednotlivých kategorii. Množina je rozšířena veškerými technikami augmentace datové základny z kapitoly 6.5. Detailní popis architektury Xception je vzhledem k rozsahu uveden v příloze 2. Síť byla učena po dobu 2500 iterací, s podmínkou časného ukončení, začne-li stoupat chyba na validační sadě, přičemž pro validační a testovací sadu je použito 10% použitých obrázků. Pro aktualizaci hodnot vah při učení je používán algoritmus AdaDelta, rychlost učení (learning rate) je stanovena na 0.001.

Při použití obrázků o rozměrech 299x299 pixelů dosahuje síť relativní přesnosti klasifikace 91%. Matice záměny v tomto případě nevykazuje průkazné tendence v záměně jednotlivých tříd. Lze však opět sledovat opakovanou záměnu mezi merlíkem a laskavcem.

		Kalasifikace neuronovou sítí									
		Bazanka Roční	Hluchavka Objímavá	Ježatka Kuří Noha	Laskavec Ohnutý	Merlík Bílý	Opletka Obecná	Violka Rolní	R	P	F-stats
Skutečná třída	Bazanka Roční	37	0	0	0	1	0	2	0.97	0.93	0.95
	Hluchavka Objímavá	0	39	0	1	0	0	0	0.98	0.95	0.96
	Ježatka Kuří Noha	0	0	39	1	0		0	0.98	1.00	0.99
	Laskavec Ohnutý	0	0	0	34	6	0	0	0.85	0.83	0.84
	Merlík Bílý	0	1	0	4	33	0	2	0.83	0.83	0.83
	Opletka Obecná	0	0	0	1	0	39	0	0.98	0.95	0.96
	Violka Rolní	3	1	0	0	0	2	34	0.85	0.89	0.87
								Celkem	0.92	0.91	0.91

Obrázek 40: Matice záměny, síť Xception

6.7.3. Architektura YOLO

Architektura YOLO slouží nejen k rozpoznání obrazu, ale také k lokalizaci objektů v obraze. Klasifikační část je pak de-facto implementací architektury VGG, rozšířená o část zaměřující se na detekci objektu. Každému objektu nalezenému v obrazu je tak přiřazena třída, spolehlivost klasifikace a střed a rozměry ohraničujícího obdélníku.

Pro učení sítě je nutné připravit množinu předem anotovaných obrázků, přičemž je důležité, aby byl v obrázku přítomen další kontext (okolí, další rostliny aj.). V tomto případě byl pro učení použit obraz o rozměrech 416x416 pixelů rozdělený do základní detekční mřížky13x13.

Architektura TinyYolo (popis vrstev v Obrázek 41) byla implementována a naučena za použití frameworku Deeplearning4j. Síť byla učena po dobu 2500 iterací, s podmínkou časného ukončení, začne-li stoupat chyba na validační sadě. Objekt byl ve výstupu označen jako detekovaný, byla-li spolehlivost detekce vyšší než 50%. Pro navržené řešení byla trénována detekční síť pomocí optimalizačního algoritmu ADAM s použitím míry učení (learning rate) 1e-3 a momentu 0.9. Váhy sítě byly inicializovány pomocí algoritmu XAVIER. Základní prvky architektury sítě jsou pak shrnuty v obrázku 41.

VertexName	(Typ vrstvy)	nIn,nOut	Tvar parametrů	InputShape	OutputShape
input_1	Vstup	-,-	-	-	-
conv2d_1	(Konvoluční Vrstva)	3,16	W:{16,3,3,3}	InputTypeConvolutional(h=416,w=416,c=3)	InputTypeConvolutional(h=416,w=416,c=16)
batch_normalization_1	(Batch Normalizace)	16,16	gamma:{1,16}, beta:{1,16}, mean:{1,16}, var:{1,16}	InputTypeConvolutional(h=416,w=416,c=16)	InputTypeConvolutional(h=416,w=416,c=16)
leaky_re_lu_1	(Leaky ReLU)	-,-	-	InputTypeConvolutional(h=416,w=416,c=16)	InputTypeConvolutional(h=416,w=416,c=16)
max_pooling2d_1	(Podvzorkovací vrstva)	-,-	-	InputTypeConvolutional(h=416,w=416,c=16)	InputTypeConvolutional(h=208,w=208,c=16)
conv2d_2	(Konvoluční Vrstva)	16,32	W:{32,16,3,3}	InputTypeConvolutional(h=208,w=208,c=16)	InputTypeConvolutional(h=208,w=208,c=32)
batch_normalization_2	(Batch Normalizace)	32,32	gamma:{1,32}, beta:{1,32}, mean:{1,32}, var:{1,32}	InputTypeConvolutional(h=208,w=208,c=32)	InputTypeConvolutional(h=208,w=208,c=32)
leaky_re_lu_2	(Leaky ReLU)	-,-	-	InputTypeConvolutional(h=208,w=208,c=32)	InputTypeConvolutional(h=208,w=208,c=32)
max_pooling2d_2	(Podvzorkovací vrstva)	-,-	-	InputTypeConvolutional(h=208,w=208,c=32)	InputTypeConvolutional(h=104,w=104,c=32)
conv2d_3	(Konvoluční Vrstva)	32,64	W:{64,32,3,3}	InputTypeConvolutional(h=104,w=104,c=32)	InputTypeConvolutional(h=104,w=104,c=64)
batch_normalization_3	(Batch Normalizace)	64,64	gamma:{1,64}, beta:{1,64}, mean:{1,64}, var:{1,64}	InputTypeConvolutional(h=104,w=104,c=64)	InputTypeConvolutional(h=104,w=104,c=64)
leaky_re_lu_3	(Leaky ReLU)	-,-	-	InputTypeConvolutional(h=104,w=104,c=64)	InputTypeConvolutional(h=104,w=104,c=64)
max_pooling2d_3	(Podvzorkovací vrstva)	-,-	-	InputTypeConvolutional(h=104,w=104,c=64)	InputTypeConvolutional(h=52,w=52,c=64)
conv2d_4	(Konvoluční Vrstva)	64,128	W:{128,64,3,3}	InputTypeConvolutional(h=52,w=52,c=64)	InputTypeConvolutional(h=52,w=52,c=128)
batch_normalization_4	(Batch Normalizace)	128,128	gamma:{1,128}, beta:{1,128}, mean:{1,128}, var:{1,128}	InputTypeConvolutional(h=52,w=52,c=128)	InputTypeConvolutional(h=52,w=52,c=128)
leaky_re_lu_4	(Leaky ReLU)	-,-	-	InputTypeConvolutional(h=52,w=52,c=128)	InputTypeConvolutional(h=52,w=52,c=128)
max_pooling2d_4	(Podvzorkovací vrstva)	-,-	-	InputTypeConvolutional(h=52,w=52,c=128)	InputTypeConvolutional(h=26,w=26,c=128)
conv2d_5	(Konvoluční Vrstva)	128,256	W:{256,128,3,3}	InputTypeConvolutional(h=26,w=26,c=128)	InputTypeConvolutional(h=26,w=26,c=256)
batch_normalization_5	(Batch Normalizace)	256,256	gamma:{1,256}, beta:{1,256}, mean:{1,256},var:{1,256}	InputTypeConvolutional(h=26,w=26,c=256)	InputTypeConvolutional(h=26,w=26,c=256)
leaky_re_lu_5	(Leaky ReLU)	-,-	-	InputTypeConvolutional(h=26,w=26,c=256)	InputTypeConvolutional(h=26,w=26,c=256)
max_pooling2d_5	(Podvzorkovací vrstva)	-,-	-	InputTypeConvolutional(h=26,w=26,c=256)	InputTypeConvolutional(h=13,w=13,c=256)
conv2d_6	(Konvoluční Vrstva)	256,512	W:{512,256,3,3}	InputTypeConvolutional(h=13,w=13,c=256)	InputTypeConvolutional(h=13,w=13,c=512)
batch_normalization_6	(Batch Normalizace)	512,512	gamma:{1,512}, beta:{1,512}, mean:{1,512}, var:{1,512}	InputTypeConvolutional(h=13,w=13,c=512)	InputTypeConvolutional(h=13,w=13,c=512)
leaky_re_lu_6	(Leaky ReLU)	-,-	-	InputTypeConvolutional(h=13,w=13,c=512)	InputTypeConvolutional(h=13,w=13,c=512)
max_pooling2d_6	(Podvzorkovací vrstva)	-,-	-	InputTypeConvolutional(h=13,w=13,c=512)	InputTypeConvolutional(h=13,w=13,c=512)
conv2d_7	(Konvoluční Vrstva)	5,121,024	W:{1024,512,3,3}	InputTypeConvolutional(h=13,w=13,c=512)	InputTypeConvolutional(h=13,w=13,c=1024)
batch_normalization_7	(Batch Normalizace)	10,241,024	gamma:{1,1024}, beta:{1,1024}, mean:{1,1024}, var:{1,1024}	InputTypeConvolutional(h=13,w=13,c=1024)	InputTypeConvolutional(h=13,w=13,c=1024)
leaky_re_lu_7	(Leaky ReLU)	-,-	-	InputTypeConvolutional(h=13,w=13,c=1024)	InputTypeConvolutional(h=13,w=13,c=1024)
conv2d_8	(Konvoluční Vrstva)	10,241,024	W:{1024,1024,3,3}	InputTypeConvolutional(h=13,w=13,c=1024)	InputTypeConvolutional(h=13,w=13,c=1024)
batch_normalization_8	(Batch Normalizace)	10,241,024	gamma:{1,1024}, beta:{1,1024}, mean:{1,1024}, var:{1,1024}	InputTypeConvolutional(h=13,w=13,c=1024)	InputTypeConvolutional(h=13,w=13,c=1024)
leaky_re_lu_8	(Leaky ReLU)	-,-	-	InputTypeConvolutional(h=13,w=13,c=1024)	InputTypeConvolutional(h=13,w=13,c=1024)
convolution2d_9	(Konvoluční Vrstva)	1024,60	W:{30,1024,1,1}	InputTypeConvolutional(h=13,w=13,c=1024)	InputTypeConvolutional(h=13,w=13,c=60)
outputs	(Výstupní vrstva)	-,-	-	InputTypeConvolutional(h=13,w=13,c=60)	InputTypeConvolutional(h=13,w=13,c=60)

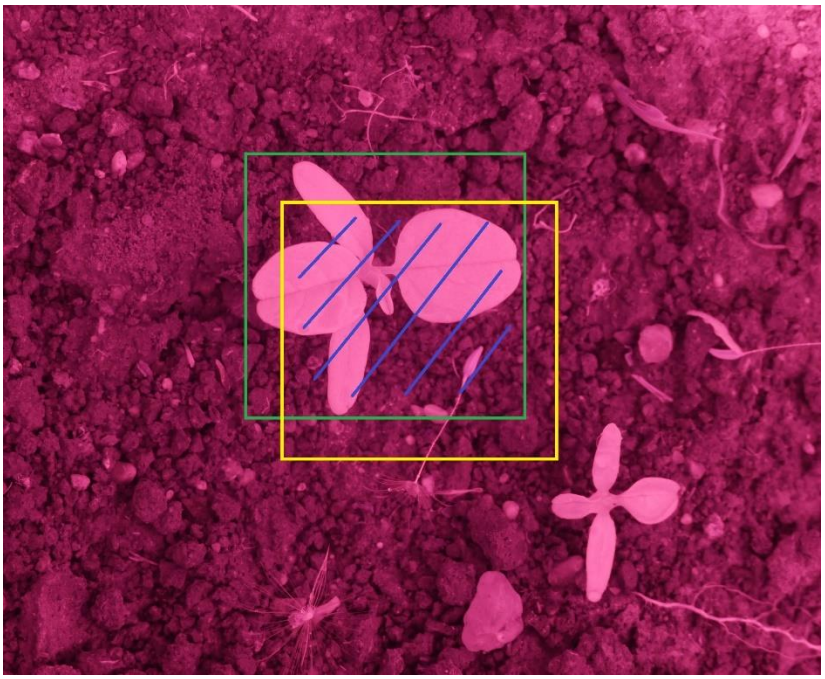
Obrázek 41: Použitá architektura TinyYolo

Při využití obrázku o rozlišení 416x416 se přesnost klasifikace pohybuje kolem 86%, přičemž matice záměny v tomto případě vyzdvihává značnou podobnost merlíku a laskavce ve velmi časných fázích růstu:

		Kalasifikace neuronovou sítí									
		Bazanka Roční	Hluchavka Objímavá	Ježatka Kuří Noha	Laskavec Ohnutý	Merlík Bílý	Opletka Obecná	Violka Rolní	R	P	F-stats
Skutečná třída	Bazanka Roční	32	0	0	0	2	0	6	0.94	0.86	0.90
	Hluchavka Objímavá	0	39	0	1	0	0	0	0.98	0.98	0.98
	Ježatka Kuří Noha	0	0	37	2	0	1	0	0.93	0.90	0.91
	Laskavec Ohnutý	1	0	0	30	8	1	0	0.75	0.79	0.77
	Merlík Bílý	0	1	4	4	30	0	1	0.75	0.75	0.75
	Opletka Obecná	0	0	0	0	0	40	0	1.00	0.91	0.95
	Violka Rolní	4	0	0	1	0	2	33	0.83	0.83	0.83
								Celkem	0.88	0.86	0.87

Obrázek 42: Matice záměny, síť YOLO

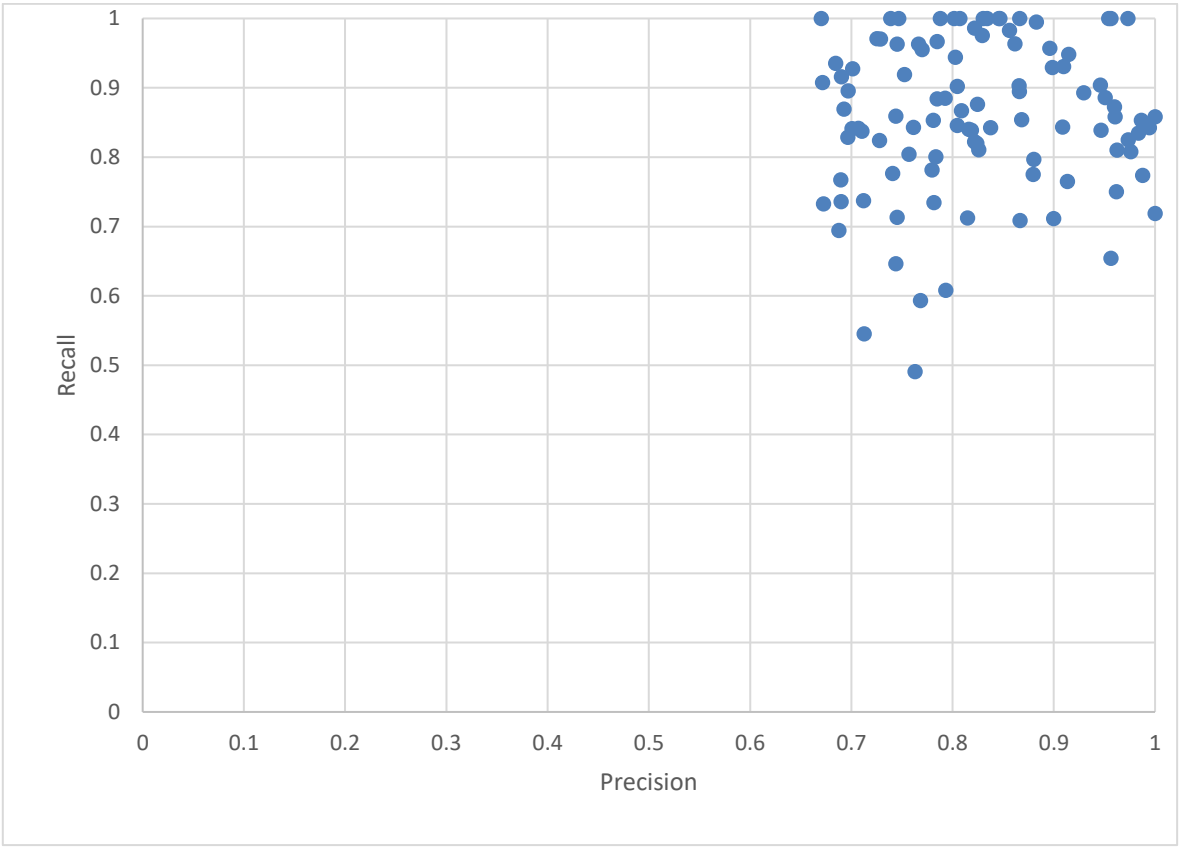
Důležitou vlastností sítí typu YOLO je schopnost určit umístění objektu v obraze. Detekce je vyhodnocena opět za použití míry precision/recall a F1 skóre. Jednotlivé veličiny jsou stanoveny dle Obrázek 43, v němž zelený obdélník představuje správně nalezený objekt a žlutý objekt nalezení sítí YOLO. Správně nalezené oblasti obrazu (TP) jsou dány oblastí průniku obou tvarů, nesprávně nalezené (FP) odpovídají části, která náleží jenom do žlutého obdélníku, zatímco nenalezené části (FN) odpovídají oblasti, která je součástí pouze zeleného obdélníku.



Obrázek 43: Hodnocení přesnosti detekce objektu, YOLO.

V případě, že jsou v jedné oblasti nalezeny 2 ohraničující obdélníky se vzájemným překryvem větším než 60% je jako třída pro danou oblast vybrán rámeček s větší spolehlivostí. Plocha obou ohraničujících obdélníků je sjednocena v případě překryvu většího než 80%.

Pro ověření bylo srovnáno 100 náhodně vybraných snímků manuálně anotovaných rostlin se snímky anotovanými sítí YOLO. Výsledná statistika precision – recall je uvedena v následujícím grafu:



Obrázek 44: Detekce YOLO: Precision-Recall

Průměrná přesnost klasifikace daná F1 skórem se pak nabývá hodnoty 0.83. Je také nutné poznamenat, že ohraničující obdélníky nemusí být při ruční anotaci stanoveny úplně přesně, a může tak docházet k situacím, kdy je výsledná detekovaná oblast uvnitř oblasti ručně anotované stanovena přesněji, ačkoliv ve výsledném srovnání dochází k poklesu hodnoty míry recall. Celkově lze konstatovat, že většina ohraničujících obdélníků detekovaných sítí YOLO ve velké míře odpovídá manuální detekci.

6.8. Souhrnné vyhodnocení

Automatické rozpoznání rostlin je významných předpokladem pro automatickou správu plevele v kulturní vegetaci. Následující tabulka prezentuje nejlepší výsledky, kterých bylo dosaženo při klasifikaci obrázku pomocí uvedených metod rozepsaných pro jednotlivé rostliny:

	CCD			Hand Crafted Features			Hu Moments			LBP			Xception			ResNet50			TinyYolo		
	R	P	F-stats	R	P	F-stats	R	P	F-stats	R	P	F-stats	R	P	F-stats	R	P	F-stats	R	P	F-stats
Bazanka Roční	0.80	0.80	**0.80**	0.93	0.62	**0.74**	0.85	0.40	**0.54**	0.71	0.60	**0.65**	0.97	0.93	0.95	0.92	0.95	0.93	0.94	0.86	**0.90**
Hluchavka Objímavá	0.50	1.00	**0.67**	0.63	0.41	**0.50**	0.48	0.25	**0.32**	0.98	0.98	**0.98**	0.98	0.95	0.96	0.98	0.95	0.96	0.98	0.98	**0.98**
Ježatka Kuří Noha	1.00	1.00	**1.00**	0.93	1.00	**0.96**	0.30	0.71	**0.42**	0.43	0.65	**0.52**	0.98	1.00	0.99	0.98	0.95	0.96	0.93	0.90	**0.91**
Laskavec Ohnutý	1.00	0.80	**0.89**	0.30	0.63	**0.41**	0.28	0.46	**0.34**	0.65	0.63	**0.64**	0.85	0.83	0.84	0.83	0.79	0.80	0.75	0.79	**0.77**
Merlík Bílý	0.20	0.31	**0.24**	0.40	0.41	**0.41**	0.15	0.38	**0.21**	0.75	0.53	**0.62**	0.83	0.83	0.83	0.78	0.79	0.78	0.75	0.75	**0.75**
Opletka Obecná	1.00	0.59	**0.74**	0.83	0.66	**0.73**	0.40	0.53	**0.46**	0.80	0.84	**0.82**	0.98	0.95	0.96	1.00	0.93	0.96	1.00	0.91	**0.95**
Violka Rolní	0.50	0.56	**0.53**	0.19	0.70	**0.30**	0.35	0.47	**0.40**	0.60	0.73	**0.66**	0.85	0.89	0.87	0.83	0.89	0.86	0.83	0.83	**0.83**
CELKEM	**0.71**	**0.72**	**0.70**	**0.60**	**0.63**	**0.58**	**0.40**	**0.45**	**0.39**	**0.70**	**0.71**	**0.70**	**0.92**	**0.91**	**0.91**	**0.90**	**0.89**	**0.90**	**0.88**	**0.86**	**0.87**

Obrázek 45: Souhrnné výsledky pro použité modely, dle rostliny

Na základě agregované podoby výsledů (tabulka 3) pak lze konstatovat, že výsledků použitelných pro reálné aplikace dosahují pouze metody založené na konvolučních neuronových sítích.

	R	P	F skóre
CCD	0.71	0.72	0.70
Hand Crafted Features	0.60	0.63	0.58
Hu Moments	0.40	0.45	0.39
LBP	0.70	0.71	0.70
TinyYOLO	0.88	0.86	0.87
ResNet50	0.90	0.89	0.90
Xception	0.92	0.91	0.91

Tabulka 3: Agregované výsledky klasifikace

6.8.1. Vliv rozměrů obrazu na výsledek

Jedním z důležitých předpokladů pro realizaci této úlohy v reálném čase a podmínkách je vyhodnotit, při jakém rozlišení je metoda rozpoznání použitelná a jaké výpočetní zdroje tedy musí být využity pro online rozpoznání plevelů. Jelikož jednotlivé konvoluční neuronové sítě vyžadují (díky své architektuře) obrázky s mírně odlišnými rozměry, byly pro následující vyhodnocení využita použita množina zmenšená na rozměr mezi 400x400 pixely a 500x500 pixely.

V tomto případě přesnost klasifikace významně klesla zejména u příznakových vektorů CCD, a příznaků Hand Crafted Features, které se ztrátou detailu v obrázku ztrácejí schopnost dostatečně přesně popsat tvar a rozlišení mezi jednotlivými druhy se stává ještě obtížnějším.

	R	P	F skóre
CCD	0.52	0.61	0.56
Hand Crafted Features	0.45	0.53	0.49
Hu Moments	0.35	0.41	0.38
LBP	0.69	0.71	0.70
TinyYOLO	0.88	0.86	0.87
ResNet50	0.90	0.89	0.90
Xception	0.92	0.91	0.91

Tabulka 4: Výsledky klasifikace při použití obrázku menšího než 500x500 pixelů

Vliv na klasifikaci pomocí operátoru LBP je naproti tomu poměrně omezený, a hlavní dopad lze přičítat zejména částečnému poškození obrazové funkce v důsledku snížení rozlišení. Konvoluční neuronové sítě jsou kvůli požadavkům na výpočetní prostředky nutné pro učení primárně využívány na obrázcích s nižším rozlišením. Vliv na výsledek tedy v těchto situacích není patrný.

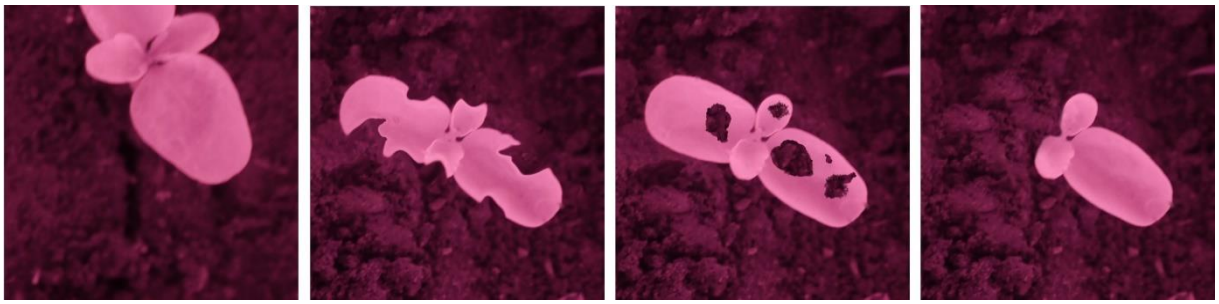
6.8.2. Vliv poškození objektu na výsledek

Přesnost klasifikace také významně klesne v případě, že by byla listová růžice výrazněji mechanicky poškozena. Pro klasifikaci rostlin v reálných polních podmínkách je však robustnost vůči takovému poškození listové růžice velmi důležitá.

V rámci práce jsou uvažována následující poškozeni:

a) Část objektu v obrazu chybí.
b) Objekt je směrem od hrany poškozený.
c) V objektu se objevují díry.
d) Část listové růžice chybí.

Každé z těchto poškození (s výjimkou odstranění celého listu z listové růžice) odstraňuje maximálně 40 procent pixelů rostliny, což je v případě manuálně vzniklých objektů ověřováno rozdílem velikosti segmentovaných objektů před poškozením a po poškození. Ukázky z množiny poškozených rostlin jsou patrné v následujícím obrázku:



Obrázek 46: Ukázky poškození objektu v obraze, zleva: chybějící část objektu, poškození hran objektu, poškození vnitřní části objektu, chybějící část objektu

Schopnost modelu klasifikovat objekt za takto stižených podmínek byla vyhodnocena za použití testovací množiny 140 obrázků, v níž je každý typ poškození rovnoměrně zastoupen napříč jednotlivými rostlinnými druhy.

Výsledná přesnost klasifikace je shrnuta v následující tabulce:

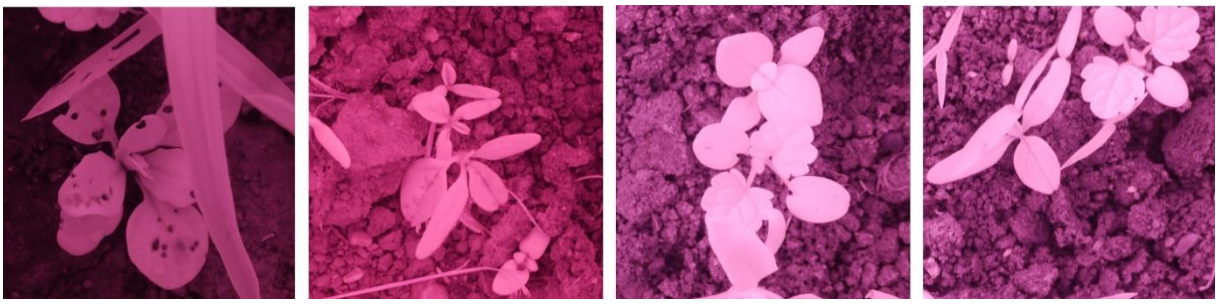
	R	P	F skóre
CCD	0.35	0.45	0.39
Hand Crafted Features	0.34	0.43	0.38
Hu Moments	0.35	0.24	0.28
LBP	0.65	0.64	0.64
TinyYOLO	0.80	0.82	0.81
ResNet50	0.88	0.83	0.85
Xception	0.78	0.81	0.79

Tabulka 5: Agregované výsledky rozpoznání poškozených rostlin

Konvoluční neuronové sítě se v tomto přístupu ukázaly jako poměrně robustní vůči poškození obrazové funkce. Největší chybovost pak vykazují objekty s významně poškozenou plochou listu směrem od hrany, které prakticky nejsou v učební množině zastoupeny. Vůči tomuto typu poškození klasifikovaného objektu je také poměrně robustní operátor LBP.

6.8.3. Vliv vzájemného překryvu rostlin

Přesnost klasifikace byla dále ověřena na množině případů, v níž dochází k částečnému překryvu plevelných rostlin a není tak vždy možné jednoznačně rozlišit jednotlivé instance objektů. Na této množině lze jednoznačně ověřit, že ačkoliv mnohé z výše uvedených příznaků (příznakových vektorů) jsou používány pro rozpoznání objektů v obrazu, jejich aplikace v reálných podmínkách by vyžadovala sofistikovanou metodu segmentace, schopnou rozlišit jednotlivé rostliny. Ukázky vzájemně se překrývajících rostlin jsou uvedeny v následujícím obrázku:



Obrázek 47: Vybrané příklady překrývajících se rostlin, zleva: violka zastíněná travinou, laskavce, bažanka obklopená plevelem, bažanka, laskavec a ježatka v jednom záběru

Je zřejmé, že metody založené na extrakci specifického příznakového vektoru jsou v těchto situacích bez použité sofistikovanější metody segmentace schopné rozlišení rostlin prakticky nepoužitelné. Jedinou výjimkou je metoda LBP, což je dáno podstatou a principem toho operátoru. Jsou-li překrývající se rostliny stejného druhu, nedochází k významné změně výsledků. Jsou-li však rostliny odlišného druhu, je výsledek obvykle dán rostlinou, na níž je vzor výraznější. Teoreticky je také možné na základě hodnot operátoru LBP provézt segmentaci obrazu. V této oblasti však nebylo dosaženo významných výsledků.

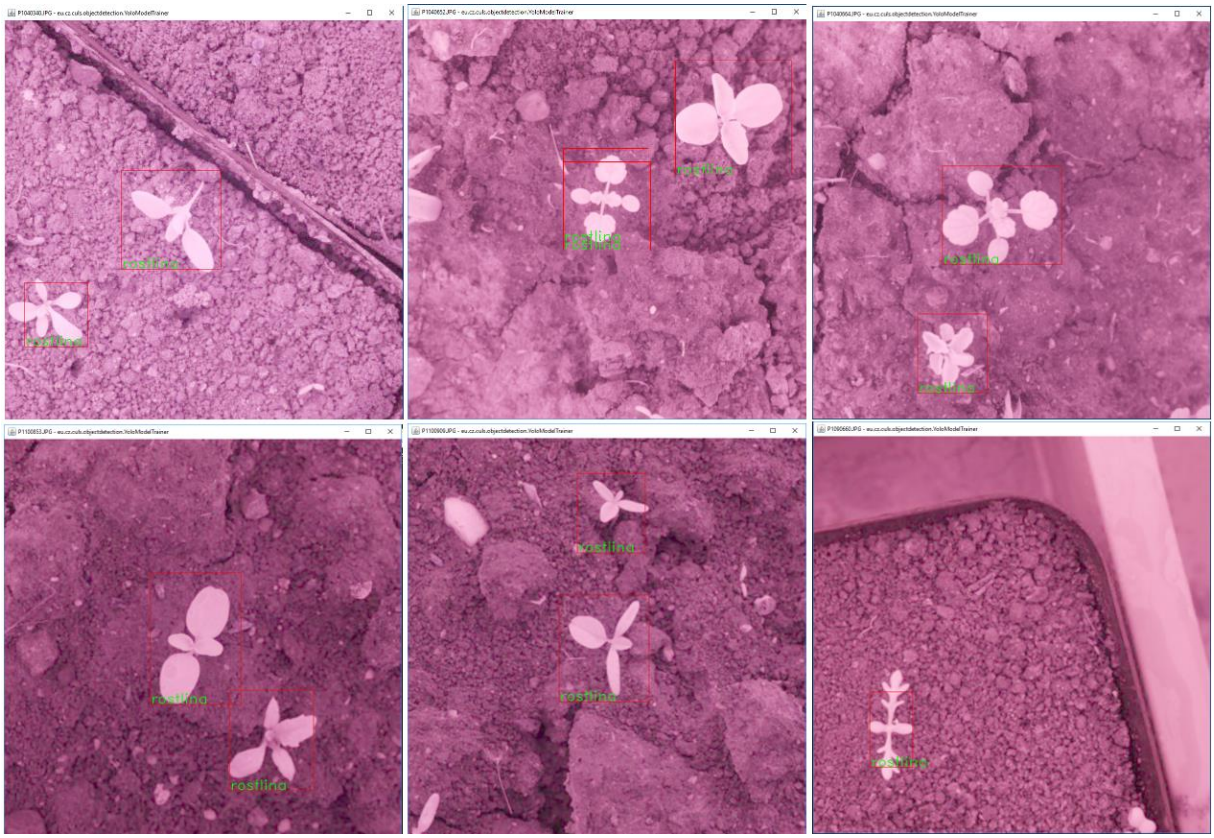
	R	P	F skóre
CCD	0.20	0.30	0.24
Hand Crafted Features	0.23	0.24	0.23
Hu Moments	0.30	0.22	0.25
LBP	0.45	0.51	0.48
TinyYOLO	0.75	0.81	0.78
ResNet50	0.56	0.65	0.60
Xception	0.60	0.69	0.64

Tabulka 6: Agregované výsledky, překryv rostlin

V případě konvolučních neuronových sítí úspěšnost také přirozené klesá. Nejlepších výsledků pak dle tabulky 6 zdaleka dosahuje síť YOLO, která v mnoha případech díky detekci úspěšně rozdělí jednotlivé části zdánlivě propojeného objektu, a provede následnou klasifikaci v jednotlivých částech obrazové scény.

6.8.4. Rozlišení listových růžic pomocí sítě YOLO a následná klasifikace

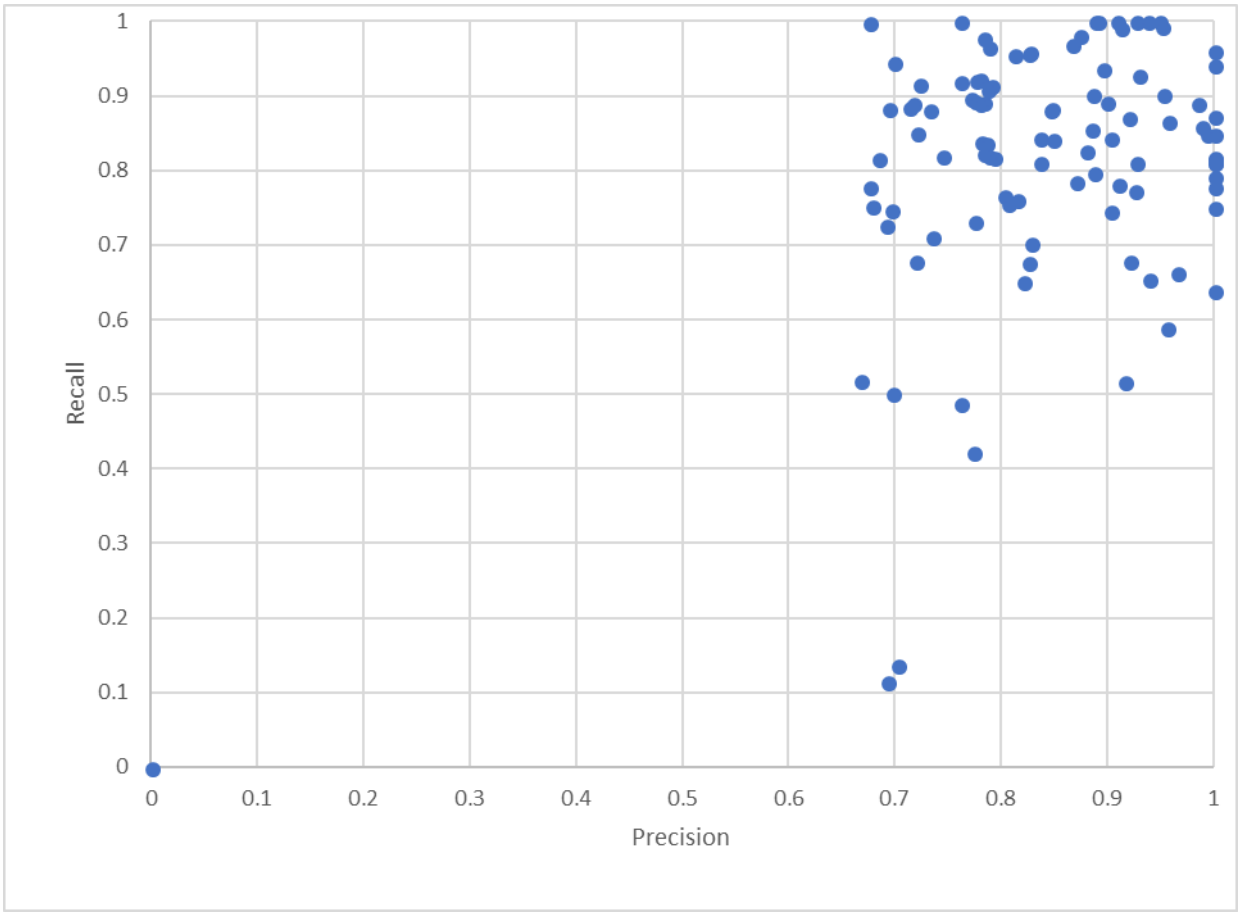
V závěrečném experimentálním přístupu byla síť YOLO použita výhradně jako mechanismus využitý pro detekci jednotlivých listových růžic. Částečně lze také tuto úpravu považovat za experiment, který ověřuje, zda je za použití větší učební množiny významně zlepšit detekční schopnosti sítě YOLO. Dílčím cílem bylo také ověřit, zda je možné zlepšit klasifikaci plevelných rostlin, je-li vybírána zájmová oblast autonomně, pomocí neuronové sítě.



Obrázek 48: Detekce plevelných rostlin pomocí sítě YOLO. Ukázka včetně detekce rostliny mimo učební sadu (v pravém dolním rohu),

Detekce plevelných rostlin byla ověřena na náhodně vybrané množině 100 anotovaných obrázků, která zahrnoval záznamy vzájemně se překrývajících rostlin (viz. předchozí kapitola). Při vyhodnocení byla použita míra spolehlivosti 0.3, byla-li spolehlivost nižší, nalezený ohraničující obdélník nebyl použit.

Při srovnání detekci s manuálně anotovanými rostlinami dosahuje síť TinyYOLO celkového F skóre 0.82. Z obrázku 49 je pak patrné, že síť v některých případech vybírá oblast větší plochy, než je oblast původně anotovaná (R=1), v dalších případech je pak detekovaný obdélník uvnitř ručně anotované oblasti (P=1). Nízká míra R pak v některých případech naznačuje, že detekce sítí YOLO vybírá pouze menší části manuálně anotované oblasti. V množině se pak objevily 3 anotované rostliny, které nebyly sítí detekovány při stanovené míře spolehlivosti. Lze předpokládat, že tyto problémy by šlo částečně odstranit větším zastoupením vzájemně se překrývajících se rostlin v učební množině. Jako druhá možnost se nabízí použití nižší míry spolehlivosti při výběru detekovaných oblastí a následným sloučením oblastí, které se vzájemně překrývají.



Obrázek 49: P/R graf detekce rostlinných růžic pomocí sítě YOLO

Zajímavým poznatkem je pak také schopnost sítě generalizovat naznačený v obrázku 52 (výřez v pravém dolním rohu), kde je síť schopna na základě příznaků extrahovaných z učební množiny označit jako rostlinu také druh, které nebyl její součástí.

	R	P	F skóre
CCD	0.28	0.35	0.31
Hand Crafted Features	0.24	0.25	0.24
Hu Moments	0.32	0.21	0.25
LBP	0.65	0.60	0.62
TinyYOLO	0.75	0.81	0.78
ResNet50	0.79	0.81	0.80
Xception	0.84	0.82	0.83

Tabulka 7: Agregované výsledky při detekci objektu sítí a následné agregaci

Oblast detekovaná sítí YOLO je následně použita pro výběr cílové oblasti, která je následně v případě metoda založených na extrakci příznakového vektoru předzpracována postupem v kapitole 6.3. Pro konvoluční neuronové sítě byla detekovaná oblast upraven a škálována tak, aby odpovídala požadovanému tvaru vstupu. Výsledná klasifikace je uvedena v tabulce 7. Při odlišení jednotlivých listových růžic významně stoupá klasifikační využitelnost metody LBP. Zároveň také stoupá schopnost sítí Xception a ResNet50 rozlišit jednotlivé druhy rostlin na úroveň, kterou lze považovat za použitelnou v praxi.

7. Metodika rozpoznání rostlin pomocí konvolučních neuronových sítí

Rozpoznání rostlin v raných fázích jejich růstu je, jak je z výsledků v předchozí kapitole patrné, netriviální úloha. I za zjednodušených podmínek lze pozorovat, že je nutné využít obrázky z poměrně vysokým rozlišením. Při poškození rostliny či vzájemném překryvu rostlin v obrazu jsou pak tradiční metody založené na extrakci předem vybraných příznaků neúčinné a jejich kvalita neodpovídá požadavkům na reálné aplikace. Pro řešení toho problému byla sestavena metodika uzpůsobená pro využití konvolučních neuronových sítí pro rozpoznání plevelných rostlin.

7.1. Strategie snímání plevelných rostlin

Dopředné neuronové sítě, jakož i konvoluční neuronové sítě využívají principu učení s učitelem (tedy *supervised learning*). Před samotným učením je tedy nutné připravit množinu vzorových příkladů, které jsou použity pro tvorbu klasifikačního modelu. Některé architektury neuronových sítí také vyžadují ruční označení a klasifikaci hledaných objektů ve scéně (např. YOLO, RCNN aj.). V případě mnoha úloh je tento přístup velmi rozumný a objekty v učební množině dokáže vybrat a označit i laik (rozpoznání značek, SPZ aj.). V jiných případech je však nutností spolupráce s expertem, který dokáže rostlinný druh z obrázku spolehlivě určit. Samotné učení je navíc obvykle časově náročné, a ne vždy je výsledná klasifikace dostatečně spolehlivá.

Řešení tohoto problému je nasnadě: cílený výsev jednotlivých druhů rostlin v kontrolovaném prostředí. Díky umělému výsevu je předem známo (s vysokou pravděpodobností, výskyt plevelů nelze nikdy vyloučit), jaká rostlina se bude v daném prostoru vyskytovat. Výhodou je také možnost ovlivňovat podmínky prostředí a simulovat tak poškození suchem, chladem či mechanické poškození.

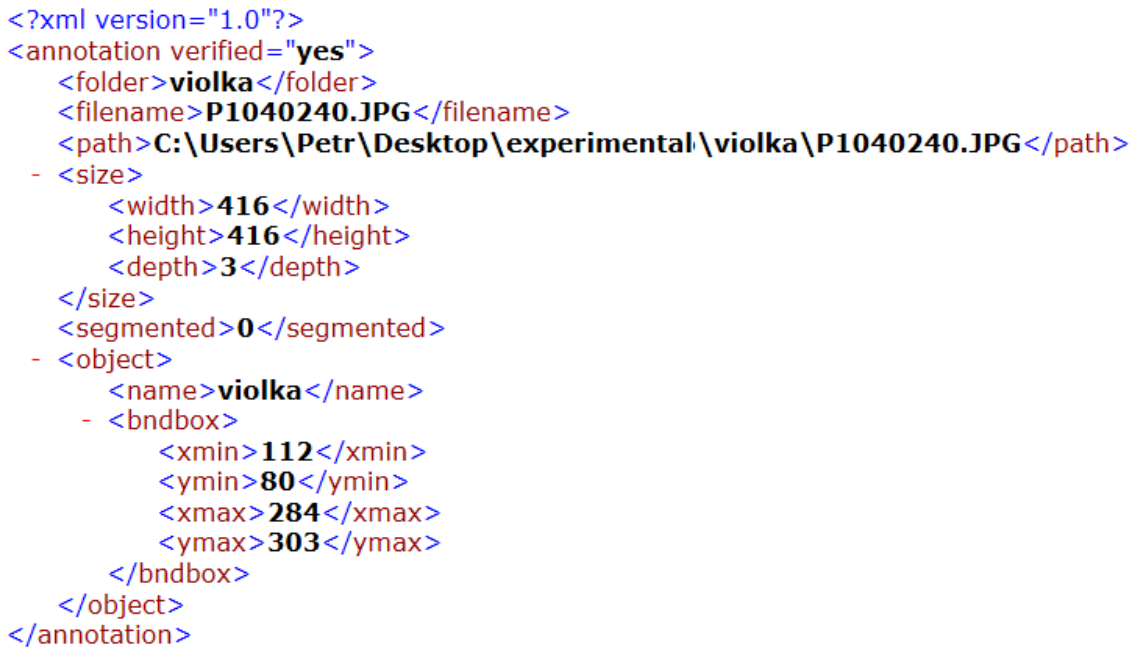
Pro vytvoření učební množiny pak postačuje rostliny nasnímat kamerou ze stabilní vzdálenosti a to takové, která je předpokládána pro budoucí aplikaci. Lze předpokládat, že ve středu fotografií pak bude objekt, na nějž se fotograf zaměřuje. V případě použití takového postupu je

možné strojové předzpracování obrazu, na jehož základě jsou označeny a vybrány oblasti obrazu použitelné jako učební sada.

7.2. Algoritmus pro automatickou anotaci objektů

Algoritmus automatické anotace rostlin ve scéně vychází z výše navrženého scénáře snímání obrazu a z výše popsaných metod pro předzpracování a segmentaci obrazu. Algoritmus automatické anotace lze shrnout následně:

1. Převod do stupňů šedi.
2. Předzpracování obrazu pomocí mediánového filtru.
3. Segmentace obrazu pomocí metody globálního prahování, hodnota prahu je vybrána metodou maximální entropie.
4. V obrazu jsou vybrány binární spojité oblasti, na základě absolutní vzdálenosti od středu scény je vybrán kandidátní objekt *O*.
5. Pro objekt *O* je metodou PCA spočtena hlavní osa, pomocí heuristiky uvedené v kapitole 6.3 jsou s objektem *O* spojeny objekty v okolí, které by měly být součástí listové růžice.
6. Pro objekt *O* je na základě maximální a minimální hodnoty x a y bodů do objektu náležejících vypočten ohraničující obdélník.
7. Na základě ohraničujícího obdélníku a složky v níž se objekt nachází je automaticky generován .xml záznam s anotací (viz. obrázek 50).



Obrázek 50: Automatická anotace objektu ve formátu XML

7.3. Využití konvoluční sítě (např. YOLO) pro detekci rostlin

Na základě množiny v anotovaných obrázků, které jsou výstupem kroku 2 uvedené metodiky lze natrénovat síť architektury YOLO. Namísto trénování sítě na rozpoznání plevelů je však množina síť YOLO natrénovat pro detekci jednotlivých listových růžic. Taková síť nahrazuje poměrně složitou a mnohdy pomalou segmentaci jednotlivých jedinců v populaci rostlin. Důležitou vlastností je pak také schopnost sítě nalézt rostliny, které jsou částečně překryté rostlinami v okolí (viz. kapitola 6.8.4).

Výhodou sítě YOLO je především rychlost, která umožňuje i detekci objektů ve videosekvenci v reálných podmínkách. Nevýhodou pak nižší přesnost klasifikace v porovnání s hlubšími modely, jejichž část pro extrakci příznaků využívá principy paralelních inception modulů a residuálních toků. Ačkoliv síť YOLO je schopná rozlišení plevelných rostlin, v současné době se jeví jako rozumnější využít tuto síť pro detekci obecnějších typů objektů (např. již zmíněných listových růžic).

Výsledkem detekce sítě jsou ohraničené oblasti v obraze, spolu s pravděpodobností výskytu plevelné rostliny. Překrývají-li se 2 takové obdélníky ve více než 80% plochy, jsou pak vzájemně sjednoceny a pro další klasifikaci je využito sjednocení těchto obdélníků.

7.4. Rozpoznání pomocí konvoluční neuronové sítě

Výsledkem detekce sítí YOLO je pozice ohraničeného obdélníku daná pozicí jeho středu a jeho rozměry. Je-li požadována vyšší přesnost klasifikace, je možné na základě toho obdélníku, resp. jeho délky (větší strany) vybrat čtvercovou oblast, která bude předložena ke klasifikaci síti, která dosahuje vyšší relativní přesnosti klasifikace. Díky tomu postupu je omezen vliv rostlin v okolí a síť je tak schopna dosahovat spolehlivých a stabilních výsledků.

Síť architektury Exeption dosahuje v rámci tohoto výzkumu v současnosti nejlepších výsledků v klasifikaci plevelných rostlin, a to i v případě poškození (kapitola 6.8.2) a v případě, že překryv jinou rostlinou omezuje náhled na základní charakteristiky použitelné pro rozpoznání rostliny (kapitola 6.8.3). Výsledkem je pak klasifikace rostliny na výřezu spolu s pravděpodobností (spolehlivostí) klasifikace vycházející z výstupní softmax vrstvy neuronové sítě. Při použití této sekvence kroků lze dosáhnout přesnosti srovnatelnou s expertní klasifikací plevelných rostlin.

7.5. Transfer learning a rozšíření na další rostliny

V první fázi konstrukce modelu schopného rozpoznání plevelných rostlin v raných fázích jejich růstu je naučena síť schopná rozpoznání specifických rostlin na základě příznaků popsaných v extrakční části architektury sítě. Tyto části jsou následně klasifikovány pomocí vícevrstvých perceptronových sítí, či jiných klasifikátorů. Vzhledem ke značné podobnosti plevelných rostlin pak lze předpokládat, že naučené příznakové mapy v konvolučních vrstvách již popisují zásadní rozpoznávací příznaky, a je tak možné znovu využít konvoluční část sítě, pro rozpoznání dalších plevelných rostlin, které nebyly součástí učební sady.

Pro využití metody transferu učení zůstává část sítě určená pro extrakci příznaků zachována, je však nutné změnit konstrukci výstupních vrstev sítě tak, aby umožňovaly klasifikaci do většího množství kategorií. Aby toto bylo realizováno, jsou odebrány neurony v klasifikační části sítě

a váhy které z nich vycházejí. Po přidání neuronů potřebných pro klasifikaci do odpovídajícího počtu kategorií jsou váhy v konvoluční části sítě zmraženy a v celé síti se pak pouze učí váhy dopředných neuronových sítí v klasifikačních vrstvách. Díky této metodě je možné rychle a s relativně nízkými náklady rozšiřovat aplikační doménu, ve které je síť použitelná.

8. Diskuze výsledků

Oblast počítačového vidění zaznamenala v poslední letech dramatických vývoj, který souvisí zejména se zvýšeným výkonem a dostupností výkonného paralelního hardwaru, především pak grafických karet, který umožnil využití a efektivní učení velmi hlubokých architektur neuronových sítí. Díky celosvětovému cílenému výzkumu pak lze pozorovat zvyšování přesnosti klasifikace na standardních datových sadách (např. CIFAR, IMAGENET, aj.) v podstatě v měsíčních intervalech.

Tato práce se namísto snahy o zlepšení výsledků na uvedených množinách zaměřuje na výzkum ve specifické oblasti rozpoznání plevelných rostlin, které je, ačkoliv se to vzhledem k ekonomickým dopadům zdá podivné, věnována poměrně malá pozornost. Namísto toho se většina rozsáhlejších výzkumu v rozpoznání rostlin zaměřuje na rozpoznání listů stromů (množiny Flavia, Herbarium aj.), případně zahradních rostlin.

V rámci výzkumu byla testována řada tradičních přístupů založených na extrakci jednotlivých atributů z tvaru rostlinné růžice. Z výsledků práce je patrné, že ačkoliv některé atributy jsou za idealizovaných podmínek vhodné pro rozpoznání, při poškození, či vzájemném překryvu přesnost výsledné klasifikace dramaticky padá. Tyto metody lze tedy teoreticky použít pouze v situacích, kdy je k dispozici velmi robustní řešení schopné rozlišení jednotlivých rostlin. I v takovém případě však absence některých částí vzniklá překryvem může vézt k devalvaci výsledné klasifikace.

Konvoluční neuronové sítě oproti tomu nabízejí řešení, které jsou robustní vůči šumům a relativně stabilní i při poškození, či překryvu jednotlivých částí rostlin. Za největší nevýhodu lze pak považovat potřebu rozsáhlých klasifikovaných množin plevelných rostlin, které v současnosti chybí nebo nejsou veřejně dostupné. Řešení úlohy je tak ve výsledku demonstrováno na relativně druhově omezené množině rostlin.

Velkou výhodou konvolučních neuronových sítí ve srovnání s tradičními přístupy je pak možnost transferu naučených příznakových map a jejich aplikace na nové, dosud nerozpoznané objekty označovaná jako transfer learning. Tato metoda je založená na využití již natrénovaných vrstev pro extrakci příznaků a následné rekonstrukci klasifikační části sítě.

Tímto způsobem je možné aplikovat již naučené příznakové mapy na úlohy rozpoznání rostlin i mimo původní učební množinu. Možností je také zobecnění na další druhy rostlin, včetně stromů, zahradních květin a dalších druhů porostů.

Úvodní kapitola této práce se zabývala pozicí úlohy rozpoznání rostlin v precizním zemědělství. Je zjevné, že pro aplikaci v reálných polních podmínkách je stále nutné dořešit řadu specifických kroků, včetně instalace snímacích zařízení na polní mechanizaci a vývoje aparátů použitelných v podmínkách, kde je častý výskyt prachu a dalších nečistot. Pro monitorování výskytu ohnisek plevelných rostlin je pak také nutné zaznamenávat geografickou polohu rostlin a vypočíst hustotu výskytu v jednotlivých oblastech. Je-li však úspěšně vyřešena specifická úloha rozpoznání jednotlivých rostlin, lze tento problém považovat za poměrně snadný – za dostačující lze považovat absolutní počet nalezených plevelných rostlin vztažený na územní jednotku. Navrhované řešení je tak dobrým základem pro konstrukci autonomních polních robotů, zaměřených na správu výskytu plevelných rostlin v kulturní vegetaci.

9. Závěr

Automatické rozpoznání rostlin je jednou z největších výzev, kterou je nutné úspěšně vyřešit pro přechod z tradičních plošných přístupů k zemědělství k zemědělství preciznímu, zejména pak v oblasti správy plevele v kulturní vegetaci.

V rámci výzkumu bylo dosaženo výsledků ve čtyřech hlavních oblastech, které odpovídají cílům této dizertační práce – byla shromážděna datová sada vhodná k řešení úlohy rozpoznání plevelných rostlin v raných fázích jejich růstu. Pomocí této sady a různých jejich variací, včetně testu pomocí snímků s omezeným rozlišením, obsahující poškozené rostliny a rostliny vzájemně se překrývající byla testována existující řešení založená na extrakci příznaků, která byla ve vybraných případech dále upravena pro lepší a přesnější klasifikaci. V reakci na rychlý rozvoj v oblasti počítačového vidění byly také využity 3 architektury konvolučních neuronových sítí, včetně architektury YOLO, která umožňuje přímou a relativně spolehlivou detekci plevelů ve scéně.

Při využití metod pro rozpoznání rostlin založených na extrakci jednoho příznakového vektoru byl ověřen očekávaný předpoklad, že mnohé metody jsou použitelné v případě, kdy jsou zkoumané rostliny nasnímány jednotlivě (odděleně) a za idealizovaných podmínek. V takovém případě lze dosáhnout přesnosti klasifikace kolem 70%, v případě LBP a rozšířeného deskriptoru CCD. Existuje-li mezi rostlinami vzájemný překryv, tyto metody často selhávají, a nejsou tak použitelné pro reálné aplikace. Oproti tomu, konvoluční neuronové sítě obsáhnou významně větší příznakový prostor, a dosahují tak lepších výsledků, v případě sítě Xception až 90%.

Na základě těchto poznatků a výsledků v kapitole 7 jsem pro řešení úlohy rozpoznání plevelných rostlin navrhl a sestavil celistvý metodický rámec využívající konvolučních neuronových sítí, detailně popsaný v kapitole 7, založený na aplikaci následujících základních kroků:

- Soubor metod pro pořízení vstupní obrazové sady vhodné pro další strojové učení založený na teoretický poznatcích metod snímání obrazu a specifických vlastnostech rostlin.

- Soubor metod předzpracování, segmentace a následní automatické anotace obrazu, který nahrazuje nutnost manuálního výběru cílových objektů pro učení neuronové sítě založená na tradičních metodách zpracování obrazu.
- Využití konvoluční neuronové sítě YOLO pro detekci pro detekci listových růžic plevelných rostlin v obraze, kterou lze teoreticky využít také pro online zpracování.
- Využití hluboké konvoluční neuronové sítě Xception pro rozpoznání jednotlivých druhů plevelných rostlin v obrazu, který využívá schopností tohoto typu architektury sítí rozpoznat objekty v obrazy s vysokou přesností.
- Využití naučených příznakových map (částí sítě určených pro extrakci příznaků) pro rozpoznání dalších druhů plevelných rostlin s využitím metody transfer learning.

Adamchuk, V. I., Hummel, J. W., Morgan, M. T. & Upadhyaya, S. K., 2004. On-the-go soil sensors for precision agriculture. *Computers and Electronics in Agriculture,* 1(44), pp. 71-91.

Aktar, M. W., Sengupta, D. & Chowdhury, A., 2009. Impact of pesticides use in agriculture: their benefits and hazards. *Interdisciplinary Toxicology,* 1(2), p. 1–12.

Al-Kufaishi, S. A., Blackmore, B. S. & Sourell, H., 2006. The feasibility of using variable rate water application under a central pivot irrigation system. *Irrigation and Drainage Systems,* Srpen, 2(20), pp. 317-327.

Andújar, D., Ribeiro, A., Fernández-Quintanilla, C. & Dorado, J., 2011. *Economic feasibility od site-specific management of Sorghum halepense in maize fields in Spain.* Prague, autor neznámý, pp. 256-263.

Arslan, S. & Colvin, T. S., 2002. Grain yield mapping: Yield sensing, yield reconstruction, and errors. *Precision Agriculture,* 2(3), pp. 135-154.

Astrand, B. & Baerveldt, A. J., 2002. An agricultural mobile robot with vision-based perception for mechanical weed control. *Autonomous Robots,* Issue 13, p. 21–35.

Auernhammer, H., 2001. Precision farming - the environmental challenge. *Computers and Electronics in Agriculture,* Issue 30, pp. 31-43.

Barghout, L. & Lee, L. W., 2003. *Perceptual information processing system.* USA, Patent č. 10/618,543.

Bay, H., Tuytelaars, T. & Van Gool, L., 2006. SURF: Speeded up Robust Features. *Computer Vision - ECCV,* Svazek 3951, pp. 404-417.

Berni, J. A., Zarco-Tejada, P. J., Suárez, L. & Fer, E., 2009. Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. *IEEE Transactions on Geoscience and Remote Sensing,* Issue 47, p. 722–738.

Biermacher, J. T. a další, 2009. Economic feasibility of site-specific optical sensing for managing nitrogen fertilizer for growing wheat. *Precision Agriculture,* 3(10), pp. 213-230.

Breiman, L., 2001. Random Forests. *Machine Learning,* 45(1), p. 5–32.

Canny, J., 1986. A Computational Approach To Edge Detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Issue 8, p. 679–698.

Cerutti, G., Tougne, L., Mille, J. & Vacavant, A., 2011. *Guiding Active Contours for Tree Leaf Segmentation and Identification.* Amsterdam, Netherlands, autor neznámý, pp. 1-15.

Cerutti, G., Tounge, L., Coquin, D. & Vacavant, A., 2014. Leaf margins as sequences: A structural approach to leaf. *Pattern Recognition Letters,* Issue 49, pp. 177-184.

Chaki, J. & Parekh, R., 2011. Plant Leaf Recognition using Shape based Features and Neural Network classifiers. *International Journal of Advanced Computer Science and Applications,* 2(10), pp. 42-47.

Chaki, J. & Parekh, R., 2012. Plant Leaf Recognition using Gabor Filter. *International Journal of Computer Applications,* 10(56), pp. 26-29.

Chaki, J., Parekh, R. & Bhattacharya, S., 2015. *Recognition of Whole and Deformed Plant Leaves using Statistical Shape Features and Neuro-Fuzzy Classifier.* Kolkata, autor neznámý, pp. 189-194.

Chaki, J. & Parekh, R., nedatováno Plant Leaf Recognition using Shape based Features and Neural Network classifiers. *International Journal of Advanced Computer Science and Applications,* 2(10), pp. 42-47.

Chen, Y.-R., Chao, K. & Kim, M. S., 2002. Machine Vision Technology for Agricultural Applications. *Computers and Electronics in Agriculture,* Issue 32, pp. 173-191.

Cho, S. I., Lee, D. S. & Jeong, J. Y., 2002. Weed–plant Discrimination by Machine Vision and Artificial Neural Network. *Biosystems Engineering,* 3(83), pp. 275-280.

Ciregan, D., Meier, U. & Schmidhuber, J., 2012. *Multi-column Deep Neural Networks for Image Classification.* Providence, RI, autor neznámý, pp. 3642-3649.

Cires¸an, D., Meier, U., Masci, J. & Schmidhuber, J., 2011. *A Committee of Neural Networks for Traffic Sign Classification.* San Jose, CA, autor neznámý, pp. 1918-1921.

Cousens, R., Brain, P., O'Donovan, J. T. & O'Sullivan, P. A., 1987. The use of biologically realistic equations to describe the effects of weed density and relative time of emergence on crop yield. *Weed Science,* Issue 35, pp. 720-725.

de Castro, A. I. a další, 2011. *Evaluation of Aerial and Quickbird images for mapping cruciferous weeds.* Prague, autor neznámý, pp. 245-255.

de Castro, A. I., Jurado-Expósito, M., Pena-Barragán, J. M. & López-Granados, F., 2012. Airborne multi-spectral imagery for mapping cruciferous weeds in cereal and legume crops. *Precision Agriculture,* 3(13), pp. 302-321.

de Castro, A. I. & López-Granados, F. J.-E. M., 2013. Broadscale cruciferous weed patch classification in winter wheat using quickbird imagery for in-season site-specific control. *Precision Agriculture,* 4(14), pp. 392-413.

Dey, V., Zhang, Y. & Zhong, M., 2010. A REVIEW ON IMAGE SEGMENTATION TECHNIQUES WITH. *ISPRS TC VII Symposium – 100 Years ISPR,* pp. 31-42.

Dieleman, J. & Mortensen, D., 1999. Characterizing the spatial pattern of Abutilon theophrasti seedling patches. *Weed Research,* Svazek 39, pp. 455-467.

El-Faki Mozib, M., Zhang, N. & Peterson, D. E., 2000. Factor affecting color-based weed detection. *Transactions of the ASAE.,* 4(43), pp. 1001-1009.

Everitt, J. H. & Villareal, R., 1987. Detecting huisache (Acacia farnesiana) and Mexican palo-verde (Parkinsonia aculeata) by aerial photography. *Weed Science,* Issue 35, pp. 427-432.

Forbes, J. C. & Watson, D., 1992. *Plants in Agriculture.* Cambridge: Cambridge University Press.

Freund, Y., 2001. An Adaptive Version of the Boost by by Majority Algorithm. *Machine Learning,* 43(3), p. 293–318.

Fukushima, K., 1980. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics,* 36(4), p. 193–202.

Geigera, F., Bengtssonb, J., Berendsea, F. & et.al, 2010. Persistent negative effects of pesticides on biodiversity and biological control potential on European farmland. *Basic and Applied Ecology*, 11(2), p. 97–105.

Gerhards, R. & Christensen, S., 2003. Real-time weed detection, decision making and patch spraying in maize, sugar beet, winter wheat and winter barley. *Weed Research,* 6(43), pp. 385-392.

Ghersa, C. M., Benech-Arnold, R. L., Satorre, E. H. & Martínez-Ghersa, M. A., 2000. Advances in weed management strategies. *Field Crops Res.*, Svazek 67, p. 95–104.

Girshick, R., 2015. *Fast R-Cnn.* místo neznámé, IEEE, p. 1440–1448.

Girshick, R., Donahue, J., Darrell, T. & Malik, J., 2016. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1), p. 142–158.

Godwin, R. & Miller, P., 2010. A review of the technologies for mapping within-field variability. *Biosystems Engineering,* 4(84), pp. 393-407.

Gómez-Casero, M. T. a další, 2011. *Classifying cruciferous weeds in cereal and legume crops using discriminant analysis.* Prague, autor neznámý, pp. 234-241.

Hahnloser, R. a další, 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature,* Svazek 405, p. 947–951.

Hall, D. a další, 2015. Evaluation of Features for Leaf Classification in Challenging Conditions. *2015 IEEE Winter Conference on Applications of Computer Vision* , pp. 797 - 804 .

Hanzlík, P., Kožíšek, F. & Pavlíček, J., 2015. Design of intelligent decision support systems in agriculture. *International Journal of Mathematics and Computers in Simulation,* Issue 9, pp. 113-118.

Haralick, R. & Shapiro, L., 1992. *Computer and Robot Vision.* 1 ed. Boston (MA): Addison-Wesley Longman Publishing Co., Inc..

Heaton, J., 2015. Encog: Library of Interchangeable Machine Learning Models for Java and C#. *Journal of Machine Learning Research,* Svazek 16, pp. 1243-1247.

He, K., Zhang , X., Ren, S. & Sun, J., 2016. *Deep Residual Learning for Image Recognition.* místo neznámé, IEEE, p. 770–778.

Hemming, J., 2000. *Computer vision for identidying weeds in crops, Dizertační práce.* Hannover: University Hannover.

Hendl, J., 2009. *Přehled statistických metod.* 3. vydání editor Praha: Portál s.r.o..

Hinton, G., Nitishsrivastava, A. & Salakhutdinov, , I. R., 2012. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. *CoRR.*

Hlaing, S. H. & Khaing, A. S., 2014. Weed and crop segmentation and classification using area thresholding. *International Journal of Research in Engineering and Technology,* 3(3), pp. 375-382.

Ho, T. K., 1995. *Random Decision Forests.* Montreal, QC, autor neznámý, p. 278–282.

Hu, M. K., 1962. Visual Pattern Recognition by Moment Invariants. *IRE Transactions on Information Theory,* 2(8), pp. 179 - 187 .

Igel, C. & Hüsken, M., 2000. *Improving the Rprop Learning Algorithm.* místo neznámé, ICSC Academic Press, pp. 115-121.

Ioffe, S. & Szegedy, C., 2015. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* Lille, France, JMLR.org, pp. 448-456.

Jafari, A., Mohtasebi, S., Jahromi, H. & Omid, M., 2006. Weed detection in sugar beet fields using machine vision. *International Journal of Agriculture and Biology,* pp. 602-605.

Jeon, H. Y., Tian, L. F. & Zhu, H., 2011. Robust Crop and Weed Segmentation under Uncontrolled Outdoor Illumination. *Sensors,* Issue 11, pp. 6270-6283.

Jeon, W.-S. & Rhee, S.-Y., 2017. Plant Leaf Recognition Using a Convolution Neural Network. *International Journal of Fuzzy Logic and Intelligent Systems,* 17(1), pp. 26-34.

Jordan, C. F., 1969. Derivation of leaf area index from qality of light on the forest floor. *Ecology*, Issue 50, pp. 663-666.

Kadir, A., Nugroho, L. E., Susanto, A. & Santosa, P. I., 2011. A Comparative Experiment of Several Shape Methods in Recognizing Plants. *International Journal of Computer Science & Information Technology (IJCSIT),* 3(3), pp. 256-263.

Kadir, A., Nugroho, L. E., Susanto, A. & Santosa, P. I., 2011. Leaf Classification Using Shape, Color, and Texture Features. *International Journal of Computer Trends and Technology*, pp. 225-230.

Karpathy, A., 2017. *CS231n Convolutional Neural Networks for Visual Recognition.* [Online] Available at: https://deeplearning4j.org/convolutionalnetwork [Přístup získán 12 6 2017].

Keeni, K., Nakayama, K. & Shimodaira, H., 1999. *Estimation of initial weights and hidden units for fast learning of multi-layer neural networks for pattern classification.* místo neznámé, IEEE, p. 1652–1656.

Kessy, A., Lewin, A. & Strimmer, K., 2015. Optimal Whitening and Decorrelation. *The American Statistician.*

Kiani, S. & Jafari, A., 2012. Crop detection and positioning in the field using discriminant analysis and neural networks based on shape features. *Journal of Agricultural Science and Technology*, Issue 14, pp. 755-765.

Kingma, D. P. & Ba, J. L., 2015. *Adam: a Method for Stochastic Optimization.*. místo neznámé, autor neznámý, p. 1–13.

Kluge, A. & Nordmeyer, H., 2009. *Automated weed detection in winter wheat by using neural networks.* Wageningen, Wageningen Academic Pub, pp. 321-327.

Krizhevsky, A., Sutskever, I. & Hinton, G., 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM,* 60(6), pp. 84-90.

Kudsk, P., 2008. Optimising herbicide dose: a straightforward approach to reduce the risk of side effects of herbicides. *The Environmentalist,* Issue 28, pp. 49-55.

Lamm, R. D., Slaughter, D. C. & Giles, D. K., 2002. Precision weed control system for cotton. *Transactions of the ASAE*, 1(45), pp. 231-238.

Larson, W. E. & Robert, P. C., 1991. Farming by soil. V: R. F. Lal, editor *Soil management for sustainability.* Ankeny, IA, USA: Soil and Water Conserv. Soc, p. 103–112.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner., P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* 86(11), pp. 2278 - 2324.

Lee, K.-B. & Hong, K.-S., 2013. An Implementation of Leaf Recognition System using Leaf Vein and Shape. *International Journal of Bio-Science and Bio-Technology*, 5(2), pp. 57-66.

Lee, S. H., Chan, C. S., Wilkin, P. & Remagnino, P., 2015. *Deep-Plant: Plant identification with convolutional neural networks.* Quebec City, QC, IEEE, pp. 452-456.

Li, C. & Tam, P., 1998. An Iterative Algorithm for Minimum Cross Entropy Thresholding. *Pattern Recognition Letters*, 18(8), pp. 771-776.

Liphadzi, K. B. & Dille, J. A., 2006. Annual weed competitiveness as affected by preemergence herbicide in corn. *Weed Sci.,* Svazek 54, p. 156–165.

Liu, Y. a další, 2007. Research on segmentation of weed images based on computer vision. *Journal of Electronics (China),* 2(24), pp. 285-288.

Long, D. S., Engel, R. E. & Siemens, M. C., 2008. Measuring grain protein concentration with in-line near infrared reflectance spectroscopy. *Agronomy Journal,* Issue 100, p. 247–252.

Long, J., Shelhamer, E. & Darrell, T., 2015. *Fully Convolutional Networks for Semantic Segmentation.* místo neznámé, autor neznámý, p. 3431–3440.

Lowe, D., 1999. Object Recognition from Local Scale-Invariant Features. *Computer Vision*, pp. 1150-1157.

Mäenpää, T., 2003. *The local binary pattern approach to texture analysis — extensions and applications (PhD. thesis).* Oulu: Infotech Oulu and Department of Electrical and Information Engineering,University of Oulu.

Mahlein, A. K. a další, 2011. *Hypespectral imaging of foilar sugar beet disease and automatic classification by the Spectral Angle Mapper algorithm.* Prague, autor neznámý, pp. 264-272.

Meloun, M. & Militký, J., 2004. *Statistická analýza experimentálních dat.* 2. editor Praha: ACADEMIA.

Meyer, G. E. a další, 1998. Textural imaging and discriminant analysis for distinguishing weeds for spot spraying. *Transactions of the ASAE,* 4(41), pp. 1189-1197.

Miller, P., 2003. Patch spraying: future role of electronics in limiting pesticide use. *Pest Management Science*, Květen, Issue 59, pp. 566-574.

Min, L., Qiang , C. & Shuicheng , Y., 2013. Network In Network. *CoRR.*

Mulla, D. J., 2013. Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps. *Biosystems Engineering, Special Issue: Sensing Technologies for Sustainable Agriculture,* 4(114), p. 358–371.

Mulla, D. J. a další, 2002. *Modeling the effect of precision agriculture: pesticide losses to surface waters.* místo neznámé, ACS, p. 304–317.

Nansen, C., Tulio, M., Swanson, R. & Weaver, D. K., 2009. Use of spatial structure analysis of hyperspectral data cubes for detection of insect-induced stress in wheat plants. *Internation Journal of Remote Sensing,* Issue 30, pp. 2447-2464.

Ng, A. a další, 2016. *UFLDL Stanford - Autoencoders.* [Online] Available at: http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/ [Přístup získán 2 3 2018].

Ng, A. a další, 2015. *UFLDL Stanford - Whitening.* [Online] Available at: http://ufldl.stanford.edu/wiki/index.php/Exercise:PCA_and_Whitening [Přístup získán 20 7 2017].

Nixon, M. S. & Aguado, A. S., 2002. *Feature Extraction and Image Processing.* Woburn: Newnes.

Oerke, E., 2006. Crop losses to pests. *Journal of Agricultural Science,* 1(144), pp. 31-43.

Ojala, T., Pietikäinen , M. & Harwood , D., 1994. Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. *Proceedings of the 12th IAPR International Conference on Pattern Recognition,* Svazek 1, pp. 582-585.

Ojala, T., Pietikäinen, M. & Mäenpää, T., 2002. Multiresolution Gray-scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence,* 24(7), pp. 971-987.

Okamoto, H., Murata, T., Kataoka, T. & Hata, S., 2007. Plant classification for weed detection using hyperspectral imaging with wavelet analysis. *Weed Biology and Management,* 7(1), pp. 31-37.

Okamoto, H., Suzuki, Y. & Noguchi, N., 2013. Field Applications of Automated Weed Control: Asia. V: S. L. Young & F. J. Pierce, editoři *Automation: The Future of Weed Control in Cropping Systems.* místo neznámé:Springer Science & Business Media, pp. 189-202.

Olshausen , B. & Field, . D., 1997. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?. *Vision Research,* 37(23), p. 3311–3325.

OpenCV, 2015. *OpenCV - Introduction to SIFT.* [Online] Available at: https://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html [Přístup získán 4 4 2017].

Otsu, N., 1979. A threshold selection method from gray-level histograms. *IEEE Trans. Sys., Man., Cyber,* p. 62–66.

Pahikkala, T. a další, 2015. Classification of Plant Species from Images of Overlapping Leaves. *Computers and Electronics in Agriculture,* C(118), pp. 186-192.

Perez, A., Lopez, F., Benlloch, J. & Christensen, S., 2000. Colour and shape analysis techniques for weed detection in cereal fields. *Computers and Electronics in Agriculture,* Issue 25, pp. 197-212.

Peters, J. a další, 2007. Random forests as a tool for ecohydrological distribution modelling. *Ecological Modelling,* 208(2-4), pp. 304-318.

Pinheiro, P., Collobert, R. & Dollár, P., 2015. *Learning to Segment Object Candidates.* místo neznámé, autor neznámý, p. 1990–1998.

Pinheiro, P., Lin, T.-Y., Collobert, R. & Dollár, P., 2016. *Learning to Refine Object Segments.* místo neznámé, Springer, p. 75–91.

Polder, G. a další, 2010. Detection of the tulip breaking virus (TBV) in tulips using optical sensors. *Precision Agriculture,* pp. 397-412.

Prewitt, J. M. S. & Mendelsohn, M. L., 1966. *Annals of the New York Academy of Sciences.* New York, New York Academy of Sciences, pp. pp. 1035-1053.

Primicerio, J. a další, 2012. A flexible unmanned aerial vehicle for precision agriculture. *Precision Agriculture,* 8, 4(13), pp. 517-523.

Qi, X., Xiao, R., Guo, J. & Zhang, L., 2012. *Pairwise rotation invariant co-occurrence local binary pattern.* místo neznámé, Springer, pp. 158-171.

Quinlan, J. R., 1986. Induction of Decision Trees. *Machine Learning,* 1(1), p. 81–106.

Quinlan, . J. R., 1993. *C4.5: Programs for Machine Learning.* San Francisco: Morgan Kaufmann.

Radclife, E. B., Hutchison, W. D. & Cancelado, R. E., 2009. *Integrated Pest Management: Concepts, Tactics, Strategies and Case Studies.* Cambridge: Cambridge University Press.

Redmon, J., Divvala, S., Girshick , R. & Farhadi, A., 2016. *You Only Look Once: Unified, Real-Time Object Detection.* místo neznámé, IEEE, p. 779–788.

Relyea, R. A., 2005. The Impact of Insecticides and Herbicides on the Biodiversity and Productivity of Aquatic Communities. *Ecological Applications,* 15(2), pp. 618-627.

Ren, S., He, K., Girshick, R. & SUN , J., 2015. *Towards Real-Time Object Detection with Region Proposal Networks.* místo neznámé, autor neznámý, p. 91–99.

Rey Otero, I. & Delbracio, M., 2014. Anatomy of the SIFT Method. *Image Processing On Line,* Svazek 4, pp. 370-396.

Riedmiller, M. & Braun, H., 1992. *RPROP - A Fast Adaptive Learning Algorithm.* místo neznámé, autor neznámý

Ritter, C. & Gerhards, R., 2007. *Can short-term gains in site-specific weed management be sustained over multiple years?.* Bonn, autor neznámý

Rouse, J. W., Haas, R. H., Shell, J. A. & Deering, D. W., 1973. *Monitoring vegetation systems in the Greant Plains with ERTS.* Washington DC, USA, autor neznámý, pp. 309-317.

Sezgin, M. & Sankur, B., 2004. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging,* 1(13), pp. 146-165.

Shapiro, L. & Stockman, G., 2002. *Computer Vision.* Prentise Hall: autor neznámý

Sheela, K. G. & Deepa, S. N., 2013. Review on Methods to Fix Number of Hidden Neurons in Neural Networks. *Mathematical Problems in Engineering,* pp. 1-11.

Shrestha, D. S., Steward, B. L. & Bartlett, E., 2001. Segmentation of plant from background using neural network approach. *Transactions on ASAE,* pp. 50-57.

Siddiqi, M. H., Ahmad, W. & Ahmad, I., 2008. Weed Classification Using Erosion and Watershed Segmentation Algorithm. *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering,* pp. 366-369.

Simonyan, K. & Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *CoRR.*

Slaughter, D. C., 2013. *The Biological Engineer: Sensing the Difference between crops and weeds.* místo neznámé, Springer Science & Business Media, pp. 71-98.

Slaughter, D. C., Giles, D. K. & Downey, D., 2008. Autonomous robotic weed control systems: a review.. *Computers and Electronics in Agriculture,* 1(61), p. 63–78.

Sun, Y., Liu, Y., Wang, G. & Zhang, H., 2017. Deep Learning for Plant Identification in Natural Environment. *Computational Intelligence and Neuroscience,* p. 6.

Szegedy, C. a další, 2015. *Going Deeper with Convolutions.* Boston, MA, IEEE, pp. 1-9.

Tang, L., Tian, L. & Steward, B., 2000. Color Image Segmentation with Genetic Algorithm for In-Field Weed Sensing. *Transactions of the ASAE,* 4(43), pp. 1019-1027.

Tang, L., Tian, L. & Steward, B. L., 2003. Classification of broadleaf and grass weeds using gabor wavelets and an artificial neural network. *Transactions of the ASAE,* 4(46), pp. 1247-1254.

Tan, K. L., Ooi, B. C. & Thiang, L. F., 2003. Retrieving similar shapes effectively and efficiently. *Multimedia Tools and Applications,* Issue 19, pp. 111-134.

Tan, K. L., Ooi, B. C. & Thiang, L. F., 2003. Retrieving similar shapes effectively and efficiently. *Multimedia Tools and Applications,* Svazek 19 , pp. 111-134.

Tian, L., 2002. Development of a sensor-based precision herbicide application system. *Computers and Electronics in Agriculture,* Issue 36, p. 133–149.

Weis, M. a další, 2008. Precision farming for weed management: techniques. *Gesunde Pflanzen,* Issue 60, pp. 171-181.

Wiles, L. J., 2005. Sampling to make maps for site-specific weed management. *Weed Science,* Issue 53, pp. 228-235.

Woebbecke, D. M., Meyer, G. E., Vonbargen, K. & Mortensen, D. A., 1995. Color indexes for weed identification under various soil, residue and lighting conditions. *Transactions on ASAE,* 1(38), pp. 259-269.

Wu, S. G. a další, 2007. *A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network.* Giza, IEEE, pp. 11-16.

Xie, S. a další, 2016. Aggregated Residual Transformations for Deep Neura. *CoRR.*

Yang, C. C., Prasher, S. O. & Landry, J. A., 2002. Weed recognition in corn fields using back-propagation neural network models. *Canadian Biosystems Engineering,* Issue 44, pp. 715-722.

Zhang, C. & Kovacs, J. M., 2012. The application of small unmanned aerial systems for precision agriculture: a review. *Precision Agriculture,* 12, 6(13), pp. 693-712.

Zhang, N., Wang, M. & Wang, N., 2002. Precision agriculture: a worldwide overview. *Computers and Electronics in Agriculture,* Issue 36, pp. 113-132.

Zhang, Y.-J., 2006. *Advances in Image and Video Segmentation.* místo neznámé:Idea Group Inc.

Zhu, H. a další, 2018. Plant identification based on very deep convolutional neural networks. *Multimedia Tools and Applications,* 77(10), p. 1–19.

Zier, P., Hank, K. & Wagner, P., 2008. Economic potential of autoguidance systems. *Berichte uber Landwirtschaft: Zeitschrift fur Agrarpolitik und Landwirtschaft,* 3(86), pp. 410-432.

Poděkování:

Děkuji celé své rodině – předně pak mamince, tatínkovi a oběma mým mladším bratřím, bez jejichž podpory bych doktorské studium nejspíš nezahájil, neabsolvoval ani nedokončil.

Srdečně děkuji svému vedoucímu, doc. Ing. Arnoštu Veselému CSc. za jeho laskavé vedení, a četné rady, bez nichž by tato práce nikdy nemohla vzniknout. Jednou bych toho chtěl o umělé inteligenci vědět tolik, co on.

Mimořádný přínos na vzniku této práce má Ing. Josefu Pavlíček Ph.D., můj kamarád a kolega. V menší, ale většinou větší míře, se spolupodílel na naprosté většině toho, co jsem během doktorského studia vytvořil, včetně softwarových knihoven, které v rámci přípravy mojí dizertační práce vznikly a včetně úvodního zprovoznění frameworků DL4J a Encog. Bez jeho přínosu a neuvěřitelné energii by obsah této práce byl výrazně skromnější.

Speciální poděkování pak věnuji Ing. Pavlu Hamouzovi, Ph.D., který je autorem fotografií plevelných rostlin, které byly použity ve výzkumu prezentovaném v této práci.

Mnohokráte děkuji svým milým přátelům, kolegům doktorandům za spolupráci a pomoc při snášení útrap doktorského studia. Zejména děkuji Ing. Ondřeji Gojdovi (doufám, že už Ph.D.) za dlouholetou vynikající a výjimečnou spolupráci i za to, že jsem si s ním mohl zahrát v kapele na kytaru. Děkuji také mojí drahé kamarádce Ing. Sylvii Kobzev-Kotáskové (doufám, že už Ph.D.) za to, kolik času se mnou na univerzitní půdě (i mimo ni) strávila, a za to že mi (teda alespoň většinou) umí zpříjemnit celý den ☺.

Srdečně děkuji také Bc. Elišce Jirmannové za mimořádnou podporu v závěrečných týdnech a dnech sepisování této práce. Pomohla mi mnohem více, než sama tuší ☺

A na závěr, čte-li někdo tuto dizertační práce až sem, když už vlastně skončila, má můj obdiv a srdečné pozvání na pivo!

Příloha 1: Architektura použité sítě ResNet

Označení vrstvy	Typ vrstvy	Počet map	Tvar parametrů	Vstupní vrstva	Tvar vstupu	Tvar výstupu
input_1	(InputVertex)	-,-	-	-	-	-
conv1	(ConvolutionLayer)	3,64	W:{64,3,7,7},b:{1,64}	[input_1]	InputTypeConvolutional(h=216,w=216,c=3)	InputTypeConvolutional(h=108,w=108,c=64)
bn_conv1	(BatchNormalization)	64,64	gamma:{1,64},beta:{1,64},mean:{1,64},var:{1,64}	[conv1]	InputTypeConvolutional(h=108,w=108,c=64)	InputTypeConvolutional(h=108,w=108,c=64)
activation_1	(ActivationLayer)	-,-	-	[bn_conv1]	InputTypeConvolutional(h=108,w=108,c=64)	InputTypeConvolutional(h=108,w=108,c=64)
max_pooling2d_1	(SubsamplingLayer)	-,-	-	[activation_1]	InputTypeConvolutional(h=108,w=108,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
res2a_branch2a	(ConvolutionLayer)	64,64	W:{64,64,1,1},b:{1,64}	[max_pooling2d_1]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
res2a_branch1	(ConvolutionLayer)	64,256	W:{256,64,1,1},b:{1,256}	[max_pooling2d_1]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=256)
bn2a_branch2a	(BatchNormalization)	64,64	gamma:{1,64},beta:{1,64},mean:{1,64},var:{1,64}	[res2a_branch2a]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
bn2a_branch1	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res2a_branch1]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=53,w=53,c=256)
activation_2	(ActivationLayer)	-,-	-	[bn2a_branch2a]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
res2a_branch2b	(ConvolutionLayer)	64,64	W:{64,64,3,3},b:{1,64}	[activation_2]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
bn2a_branch2b	(BatchNormalization)	64,64	gamma:{1,64},beta:{1,64},mean:{1,64},var:{1,64}	[res2a_branch2b]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
activation_3	(ActivationLayer)	-,-	-	[bn2a_branch2b]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
res2a_branch2c	(ConvolutionLayer)	64,256	W:{256,64,1,1},b:{1,256}	[activation_3]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=256)
bn2a_branch2c	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res2a_branch2c]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=53,w=53,c=256)
add_1	(ElementWiseVertex)	-,-	-	[bn2a_branch2c,bn2a_branch1]	-	InputTypeConvolutional(h=53,w=53,c=256)
activation_4	(ActivationLayer)	-,-	-	[add_1]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=53,w=53,c=256)
res2b_branch2a	(ConvolutionLayer)	256,64	W:{64,256,1,1},b:{1,64}	[activation_4]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=53,w=53,c=64)
bn2b_branch2a	(BatchNormalization)	64,64	gamma:{1,64},beta:{1,64},mean:{1,64},var:{1,64}	[res2b_branch2a]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
activation_5	(ActivationLayer)	-,-	-	[bn2b_branch2a]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
res2b_branch2b	(ConvolutionLayer)	64,64	W:{64,64,3,3},b:{1,64}	[activation_5]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)

bn2b_branch2b	(BatchNormalization)	64,64	gamma:{1,64},beta:{1,64},mean:{1,64},var:{1,64}	[res2b_branch2b]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
activation_6	(ActivationLayer)	-,-	-	[bn2b_branch2b]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
res2b_branch2c	(ConvolutionLayer)	64,256	W:{256,64,1,1},b:{1,256}	[activation_6]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=256)
bn2b_branch2c	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res2b_branch2c]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=53,w=53,c=256)
add_2	(ElementWiseVertex)	-,-	-	[bn2b_branch2c,activation_4]	-	InputTypeConvolutional(h=53,w=53,c=256)
activation_7	(ActivationLayer)	-,-	-	[add_2]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=53,w=53,c=256)
res2c_branch2a	(ConvolutionLayer)	256,64	W:{64,256,1,1},b:{1,64}	[activation_7]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=53,w=53,c=64)
bn2c_branch2a	(BatchNormalization)	64,64	gamma:{1,64},beta:{1,64},mean:{1,64},var:{1,64}	[res2c_branch2a]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
activation_8	(ActivationLayer)	-,-	-	[bn2c_branch2a]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
res2c_branch2b	(ConvolutionLayer)	64,64	W:{64,64,3,3},b:{1,64}	[activation_8]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
bn2c_branch2b	(BatchNormalization)	64,64	gamma:{1,64},beta:{1,64},mean:{1,64},var:{1,64}	[res2c_branch2b]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
activation_9	(ActivationLayer)	-,-	-	[bn2c_branch2b]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=64)
res2c_branch2c	(ConvolutionLayer)	64,256	W:{256,64,1,1},b:{1,256}	[activation_9]	InputTypeConvolutional(h=53,w=53,c=64)	InputTypeConvolutional(h=53,w=53,c=256)
bn2c_branch2c	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res2c_branch2c]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=53,w=53,c=256)
add_3	(ElementWiseVertex)	-,-	-	[bn2c_branch2c,activation_7]	-	InputTypeConvolutional(h=53,w=53,c=256)
activation_10	(ActivationLayer)	-,-	-	[add_3]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=53,w=53,c=256)
res3a_branch2a	(ConvolutionLayer)	256,128	W:{128,256,1,1},b:{1,128}	[activation_10]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=27,w=27,c=128)
res3a_branch1	(ConvolutionLayer)	256,512	W:{512,256,1,1},b:{1,512}	[activation_10]	InputTypeConvolutional(h=53,w=53,c=256)	InputTypeConvolutional(h=27,w=27,c=512)
bn3a_branch2a	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[res3a_branch2a]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
bn3a_branch1	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res3a_branch1]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=512)
activation_11	(ActivationLayer)	-,-	-	[bn3a_branch2a]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
res3a_branch2b	(ConvolutionLayer)	128,128	W:{128,128,3,3},b:{1,128}	[activation_11]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
bn3a_branch2b	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[res3a_branch2b]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
activation_12	(ActivationLayer)	-,-	-	[bn3a_branch2b]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
res3a_branch2c	(ConvolutionLayer)	128,512	W:{512,128,1,1},b:{1,512}	[activation_12]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=512)
bn3a_branch2c	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res3a_branch2c]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=512)

add_4	(ElementWiseVertex)	-,-	-	[bn3a_branch2c,bn3a_branch1]	-	InputTypeConvolutional(h=27,w=27,c=512)
activation_13	(ActivationLayer)	-,-	-	[add_4]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=512)
res3b_branch2a	(ConvolutionLayer)	512,128	W:{128,512,1,1},b:{1,128}	[activation_13]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=128)
bn3b_branch2a	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[res3b_branch2a]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
activation_14	(ActivationLayer)	-,-	-	[bn3b_branch2a]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
res3b_branch2b	(ConvolutionLayer)	128,128	W:{128,128,3,3},b:{1,128}	[activation_14]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
bn3b_branch2b	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[res3b_branch2b]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
activation_15	(ActivationLayer)	-,-	-	[bn3b_branch2b]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
res3b_branch2c	(ConvolutionLayer)	128,512	W:{512,128,1,1},b:{1,512}	[activation_15]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=512)
bn3b_branch2c	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res3b_branch2c]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=512)
add_5	(ElementWiseVertex)	-,-	-	[bn3b_branch2c,activation_13]		InputTypeConvolutional(h=27,w=27,c=512)
activation_16	(ActivationLayer)	-,-	-	[add_5]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=512)
res3c_branch2a	(ConvolutionLayer)	512,128	W:{128,512,1,1},b:{1,128}	[activation_16]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=128)
bn3c_branch2a	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[res3c_branch2a]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
activation_17	(ActivationLayer)	-,-	-	[bn3c_branch2a]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
res3c_branch2b	(ConvolutionLayer)	128,128	W:{128,128,3,3},b:{1,128}	[activation_17]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
bn3c_branch2b	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[res3c_branch2b]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
activation_18	(ActivationLayer)	-,-	-	[bn3c_branch2b]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
res3c_branch2c	(ConvolutionLayer)	128,512	W:{512,128,1,1},b:{1,512}	[activation_18]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=512)
bn3c_branch2c	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res3c_branch2c]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=512)
add_6	(ElementWiseVertex)	-,-	-	[bn3c_branch2c,activation_16]	-	InputTypeConvolutional(h=27,w=27,c=512)
activation_19	(ActivationLayer)	-,-	-	[add_6]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=512)
res3d_branch2a	(ConvolutionLayer)	512,128	W:{128,512,1,1},b:{1,128}	[activation_19]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=128)
bn3d_branch2a	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[res3d_branch2a]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
activation_20	(ActivationLayer)	-,-	-	[bn3d_branch2a]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
res3d_branch2b	(ConvolutionLayer)	128,128	W:{128,128,3,3},b:{1,128}	[activation_20]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)

bn3d_branch2b	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[res3d_branch2b]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
activation_21	(ActivationLayer)	-,-	-	[bn3d_branch2b]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=128)
res3d_branch2c	(ConvolutionLayer)	128,512	W:{512,128,1,1},b:{1,512}	[activation_21]	InputTypeConvolutional(h=27,w=27,c=128)	InputTypeConvolutional(h=27,w=27,c=512)
bn3d_branch2c	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res3d_branch2c]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=512)
add_7	(ElementWiseVertex)	-,-	-	[bn3d_branch2c,activation_19]	-	InputTypeConvolutional(h=27,w=27,c=512)
activation_22	(ActivationLayer)	-,-	-	[add_7]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=27,w=27,c=512)
res4a_branch1	(ConvolutionLayer)	512,1024	W:{1024,512,1,1},b:{1,1024}	[activation_22]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=14,w=14,c=1024)
res4a_branch2a	(ConvolutionLayer)	512,256	W:{256,512,1,1},b:{1,256}	[activation_22]	InputTypeConvolutional(h=27,w=27,c=512)	InputTypeConvolutional(h=14,w=14,c=256)
bn4a_branch1	(BatchNormalization)	1024,1024	gamma:{1,1024},beta:{1,1024},mean:{1,1024},var:{1,1024}	[res4a_branch1]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
bn4a_branch2a	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4a_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_23	(ActivationLayer)	-,-	-	[bn4a_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4a_branch2b	(ConvolutionLayer)	256,256	W:{256,256,3,3},b:{1,256}	[activation_23]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
bn4a_branch2b	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4a_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_24	(ActivationLayer)	-,-	-	[bn4a_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4a_branch2c	(ConvolutionLayer)	256,1024	W:{1024,256,1,1},b:{1,1024}	[activation_24]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=1024)
bn4a_branch2c	(BatchNormalization)	1024,1024	gamma:{1,1024},beta:{1,1024},mean:{1,1024},var:{1,1024}	[res4a_branch2c]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
add_8	(ElementWiseVertex)	-,-	-	[bn4a_branch2c,bn4a_branch1]	-	InputTypeConvolutional(h=14,w=14,c=1024)
activation_25	(ActivationLayer)	-,-	-	[add_8]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
res4b_branch2a	(ConvolutionLayer)	1024,256	W:{256,1024,1,1},b:{1,256}	[activation_25]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=256)
bn4b_branch2a	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4b_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_26	(ActivationLayer)	-,-	-	[bn4b_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4b_branch2b	(ConvolutionLayer)	256,256	W:{256,256,3,3},b:{1,256}	[activation_26]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
bn4b_branch2b	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4b_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_27	(ActivationLayer)	-,-	-	[bn4b_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4b_branch2c	(ConvolutionLayer)	256,1024	W:{1024,256,1,1},b:{1,1024}	[activation_27]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=1024)
bn4b_branch2c	(BatchNormalization)	1024,1024	gamma:{1,1024},beta:{1,1024},mean:{1,1024},var:{1,1024}	[res4b_branch2c]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)

add_9	(ElementWiseVertex)	-,-	-	[bn4b_branch2c,activation_25]	-	InputTypeConvolutional(h=14,w=14,c=1024)
activation_28	(ActivationLayer)	-,-	-	[add_9]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
res4c_branch2a	(ConvolutionLayer)	1024,256	W:{256,1024,1,1},b:{1,256}	[activation_28]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=256)
bn4c_branch2a	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4c_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_29	(ActivationLayer)	-,-	-	[bn4c_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4c_branch2b	(ConvolutionLayer)	256,256	W:{256,256,3,3},b:{1,256}	[activation_29]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
bn4c_branch2b	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4c_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_30	(ActivationLayer)	-,-	-	[bn4c_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4c_branch2c	(ConvolutionLayer)	256,1024	W:{1024,256,1,1},b:{1,1024}	[activation_30]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=1024)
bn4c_branch2c	(BatchNormalization)	1024,1024	gamma:{1,1024},beta:{1,1024},mean:{1,1024},var:{1,1024}	[res4c_branch2c]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
add_10	(ElementWiseVertex)	-,-	-	[bn4c_branch2c,activation_28]	-	InputTypeConvolutional(h=14,w=14,c=1024)
activation_31	(ActivationLayer)	-,-	-	[add_10]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
res4d_branch2a	(ConvolutionLayer)	1024,256	W:{256,1024,1,1},b:{1,256}	[activation_31]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=256)
bn4d_branch2a	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4d_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_32	(ActivationLayer)	-,-	-	[bn4d_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4d_branch2b	(ConvolutionLayer)	256,256	W:{256,256,3,3},b:{1,256}	[activation_32]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
bn4d_branch2b	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4d_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_33	(ActivationLayer)	-,-	-	[bn4d_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4d_branch2c	(ConvolutionLayer)	256,1024	W:{1024,256,1,1},b:{1,1024}	[activation_33]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=1024)
bn4d_branch2c	(BatchNormalization)	1024,1024	gamma:{1,1024},beta:{1,1024},mean:{1,1024},var:{1,1024}	[res4d_branch2c]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
add_11	(ElementWiseVertex)	-,-	-	[bn4d_branch2c,activation_31]	-	InputTypeConvolutional(h=14,w=14,c=1024)
activation_34	(ActivationLayer)	-,-	-	[add_11]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
res4e_branch2a	(ConvolutionLayer)	1024,256	W:{256,1024,1,1},b:{1,256}	[activation_34]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=256)
bn4e_branch2a	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4e_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_35	(ActivationLayer)	-,-	-	[bn4e_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4e_branch2b	(ConvolutionLayer)	256,256	W:{256,256,3,3},b:{1,256}	[activation_35]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)

bn4e_branch2b	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4e_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_36	(ActivationLayer)	-,-	-	[bn4e_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4e_branch2c	(ConvolutionLayer)	256,1024	W:{1024,256,1,1},b:{1,1024}	[activation_36]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=1024)
bn4e_branch2c	(BatchNormalization)	1024,1024	gamma:{1,1024},beta:{1,1024},mean:{1,1024},var:{1,1024}	[res4e_branch2c]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
add_12	(ElementWiseVertex)	-,-	-	[bn4e_branch2c,activation_34]	-	InputTypeConvolutional(h=14,w=14,c=1024)
activation_37	(ActivationLayer)	-,-	-	[add_12]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
res4f_branch2a	(ConvolutionLayer)	1024,256	W:{256,1024,1,1},b:{1,256}	[activation_37]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=256)
bn4f_branch2a	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4f_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_38	(ActivationLayer)	-,-	-	[bn4f_branch2a]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4f_branch2b	(ConvolutionLayer)	256,256	W:{256,256,3,3},b:{1,256}	[activation_38]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
bn4f_branch2b	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[res4f_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
activation_39	(ActivationLayer)	-,-	-	[bn4f_branch2b]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=256)
res4f_branch2c	(ConvolutionLayer)	256,1024	W:{1024,256,1,1},b:{1,1024}	[activation_39]	InputTypeConvolutional(h=14,w=14,c=256)	InputTypeConvolutional(h=14,w=14,c=1024)
bn4f_branch2c	(BatchNormalization)	1024,1024	gamma:{1,1024},beta:{1,1024},mean:{1,1024},var:{1,1024}	[res4f_branch2c]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
add_13	(ElementWiseVertex)	-,-	-	[bn4f_branch2c,activation_37]	-	InputTypeConvolutional(h=14,w=14,c=1024)
activation_40	(ActivationLayer)	-,-	-	[add_13]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=14,w=14,c=1024)
res5a_branch1	(ConvolutionLayer)	1024,2048	W:{2048,1024,1,1},b:{1,2048}	[activation_40]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=7,w=7,c=2048)
res5a_branch2a	(ConvolutionLayer)	1024,512	W:{512,1024,1,1},b:{1,512}	[activation_40]	InputTypeConvolutional(h=14,w=14,c=1024)	InputTypeConvolutional(h=7,w=7,c=512)
bn5a_branch1	(BatchNormalization)	2048,2048	gamma:{1,2048},beta:{1,2048},mean:{1,2048},var:{1,2048}	[res5a_branch1]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=7,w=7,c=2048)
bn5a_branch2a	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res5a_branch2a]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
activation_41	(ActivationLayer)	-,-	-	[bn5a_branch2a]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
res5a_branch2b	(ConvolutionLayer)	512,512	W:{512,512,3,3},b:{1,512}	[activation_41]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
bn5a_branch2b	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res5a_branch2b]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
activation_42	(ActivationLayer)	-,-	-	[bn5a_branch2b]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
res5a_branch2c	(ConvolutionLayer)	512,2048	W:{2048,512,1,1},b:{1,2048}	[activation_42]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=2048)
bn5a_branch2c	(BatchNormalization)	2048,2048	gamma:{1,2048},beta:{1,2048},mean:{1,2048},var:{1,2048}	[res5a_branch2c]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=7,w=7,c=2048)

add_14	(ElementWiseVertex)	-,-	-	[bn5a_branch2c,bn5a_branch1]	-	InputTypeConvolutional(h=7,w=7,c=2048)
activation_43	(ActivationLayer)	-,-	-	[add_14]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=7,w=7,c=2048)
res5b_branch2a	(ConvolutionLayer)	2048,512	W:{512,2048,1,1},b:{1,512}	[activation_43]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=7,w=7,c=512)
bn5b_branch2a	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res5b_branch2a]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
activation_44	(ActivationLayer)	-,-	-	[bn5b_branch2a]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
res5b_branch2b	(ConvolutionLayer)	512,512	W:{512,512,3,3},b:{1,512}	[activation_44]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
bn5b_branch2b	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res5b_branch2b]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
activation_45	(ActivationLayer)	-,-	-	[bn5b_branch2b]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
res5b_branch2c	(ConvolutionLayer)	512,2048	W:{2048,512,1,1},b:{1,2048}	[activation_45]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=2048)
bn5b_branch2c	(BatchNormalization)	2048,2048	gamma:{1,2048},beta:{1,2048},mean:{1,2048},var:{1,2048}	[res5b_branch2c]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=7,w=7,c=2048)
add_15	(ElementWiseVertex)	-,-	-	[bn5b_branch2c,activation_43]	-	InputTypeConvolutional(h=7,w=7,c=2048)
activation_46	(ActivationLayer)	-,-	-	[add_15]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=7,w=7,c=2048)
res5c_branch2a	(ConvolutionLayer)	2048,512	W:{512,2048,1,1},b:{1,512}	[activation_46]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=7,w=7,c=512)
bn5c_branch2a	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res5c_branch2a]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
activation_47	(ActivationLayer)	-,-	-	[bn5c_branch2a]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
res5c_branch2b	(ConvolutionLayer)	512,512	W:{512,512,3,3},b:{1,512}	[activation_47]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
bn5c_branch2b	(BatchNormalization)	512,512	gamma:{1,512},beta:{1,512},mean:{1,512},var:{1,512}	[res5c_branch2b]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
activation_48	(ActivationLayer)	-,-	-	[bn5c_branch2b]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=512)
res5c_branch2c	(ConvolutionLayer)	512,2048	W:{2048,512,1,1},b:{1,2048}	[activation_48]	InputTypeConvolutional(h=7,w=7,c=512)	InputTypeConvolutional(h=7,w=7,c=2048)
bn5c_branch2c	(BatchNormalization)	2048,2048	gamma:{1,2048},beta:{1,2048},mean:{1,2048},var:{1,2048}	[res5c_branch2c]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=7,w=7,c=2048)
add_16	(ElementWiseVertex)	-,-	-	[bn5c_branch2c,activation_46]	-	InputTypeConvolutional(h=7,w=7,c=2048)
activation_49	(ActivationLayer)	-,-	-	[add_16]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=7,w=7,c=2048)
avg_pool	(SubsamplingLayer)	-,-	-	[activation_49]	InputTypeConvolutional(h=7,w=7,c=2048)	InputTypeConvolutional(h=1,w=1,c=2048)
flatten_1	(PreprocessorVertex)	-,-	-	[avg_pool]	-	InputTypeFeedForward(2048)
fc1000	(DenseLayer)	2048,1000	W:{2048,1000},b:{1,1000}	[flatten_1]	InputTypeFeedForward(2048)	InputTypeFeedForward(1000)

Příloha 2: Architektura použité sítě Xception

Označení vrstvy	Typ vrstvy	Počet map	Tvar parametrů	Vstupni vrstva	Tvar vstupu	Tvar výstupu
input_3	(InputVertex)	-,-	-	-	-	-
block1_conv1	(ConvolutionLayer)	3,32	W:{32,3,3,3}	[input_3]	InputTypeConvolutional(h=299,w=299,c=3)	InputTypeConvolutional(h=149,w=149,c=32)
block1_conv1_bn	(BatchNormalization)	32,32	gamma:{1,32},beta:{1,32},mean:{1,32},var:{1,32}	[block1_conv1]	InputTypeConvolutional(h=149,w=149,c=32)	InputTypeConvolutional(h=149,w=149,c=32)
block1_conv1_act	(ActivationLayer)	-,-	-	[block1_conv1_bn]	InputTypeConvolutional(h=149,w=149,c=32)	InputTypeConvolutional(h=149,w=149,c=32)
block1_conv2	(ConvolutionLayer)	32,64	W:{64,32,3,3}	[block1_conv1_act]	InputTypeConvolutional(h=149,w=149,c=32)	InputTypeConvolutional(h=147,w=147,c=64)
block1_conv2_bn	(BatchNormalization)	64,64	gamma:{1,64},beta:{1,64},mean:{1,64},var:{1,64}	[block1_conv2]	InputTypeConvolutional(h=147,w=147,c=64)	InputTypeConvolutional(h=147,w=147,c=64)
block1_conv2_act	(ActivationLayer)	-,-	-	[block1_conv2_bn]	InputTypeConvolutional(h=147,w=147,c=64)	InputTypeConvolutional(h=147,w=147,c=64)
block2_sepconv1	(SeparableConvolution2DLayer)	64,128	W:{1,64,3,3},pW:{128,64,1,1}	[block1_conv2_act]	InputTypeConvolutional(h=147,w=147,c=64)	InputTypeConvolutional(h=147,w=147,c=128)
conv2d_5	(ConvolutionLayer)	64,128	W:{128,64,1,1}	[block1_conv2_act]	InputTypeConvolutional(h=147,w=147,c=64)	InputTypeConvolutional(h=74,w=74,c=128)
block2_sepconv1_bn	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[block2_sepconv1]	InputTypeConvolutional(h=147,w=147,c=128)	InputTypeConvolutional(h=147,w=147,c=128)
batch_normalization_5	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[conv2d_5]	InputTypeConvolutional(h=74,w=74,c=128)	InputTypeConvolutional(h=74,w=74,c=128)
block2_sepconv2_act	(ActivationLayer)	-,-	-	[block2_sepconv1_bn]	InputTypeConvolutional(h=147,w=147,c=128)	InputTypeConvolutional(h=147,w=147,c=128)
block2_sepconv2	(SeparableConvolution2DLayer)	128,128	W:{1,128,3,3},pW:{128,128,1,1}	[block2_sepconv2_act]	InputTypeConvolutional(h=147,w=147,c=128)	InputTypeConvolutional(h=147,w=147,c=128)
block2_sepconv2_bn	(BatchNormalization)	128,128	gamma:{1,128},beta:{1,128},mean:{1,128},var:{1,128}	[block2_sepconv2]	InputTypeConvolutional(h=147,w=147,c=128)	InputTypeConvolutional(h=147,w=147,c=128)
block2_pool	(SubsamplingLayer)	-,-	-	[block2_sepconv2_bn]	InputTypeConvolutional(h=147,w=147,c=128)	InputTypeConvolutional(h=74,w=74,c=128)
add_29	(ElementWiseVertex)	-,-	-	[block2_pool,batch_normalization_5]	-	InputTypeConvolutional(h=74,w=74,c=128)
block3_sepconv1_act	(ActivationLayer)	-,-	-	[add_29]	InputTypeConvolutional(h=74,w=74,c=128)	InputTypeConvolutional(h=74,w=74,c=128)
conv2d_6	(ConvolutionLayer)	128,256	W:{256,128,1,1}	[add_29]	InputTypeConvolutional(h=74,w=74,c=128)	InputTypeConvolutional(h=37,w=37,c=256)

block3_sepconv1	(SeparableConvolution2DLayer)	128,256	W:{1,128,3,3},pW:{256,128,1,1}	[block3_sepconv1_act]	InputTypeConvolutional(h=74,w=74,c=128)	InputTypeConvolutional(h=74,w=74,c=256)
batch_normalization_6	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[conv2d_6]	InputTypeConvolutional(h=37,w=37,c=256)	InputTypeConvolutional(h=37,w=37,c=256)
block3_sepconv1_bn	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[block3_sepconv1]	InputTypeConvolutional(h=74,w=74,c=256)	InputTypeConvolutional(h=74,w=74,c=256)
block3_sepconv2_act	(ActivationLayer)	-,-	-	[block3_sepconv1_bn]	InputTypeConvolutional(h=74,w=74,c=256)	InputTypeConvolutional(h=74,w=74,c=256)
block3_sepconv2	(SeparableConvolution2DLayer)	256,256	W:{1,256,3,3},pW:{256,256,1,1}	[block3_sepconv2_act]	InputTypeConvolutional(h=74,w=74,c=256)	InputTypeConvolutional(h=74,w=74,c=256)
block3_sepconv2_bn	(BatchNormalization)	256,256	gamma:{1,256},beta:{1,256},mean:{1,256},var:{1,256}	[block3_sepconv2]	InputTypeConvolutional(h=74,w=74,c=256)	InputTypeConvolutional(h=74,w=74,c=256)
block3_pool	(SubsamplingLayer)	-,-	-	[block3_sepconv2_bn]	InputTypeConvolutional(h=74,w=74,c=256)	InputTypeConvolutional(h=37,w=37,c=256)
add_30	(ElementWiseVertex)	-,-	-	[block3_pool,batch_normalization_6]	InputTypeConvolutional(h=37,w=37,c=256)	InputTypeConvolutional(h=37,w=37,c=256)
conv2d_7	(ConvolutionLayer)	256,728	W:{728,256,1,1}	[add_30]	InputTypeConvolutional(h=37,w=37,c=256)	InputTypeConvolutional(h=19,w=19,c=728)
block4_sepconv1_act	(ActivationLayer)	-,-	-	[add_30]	InputTypeConvolutional(h=37,w=37,c=256)	InputTypeConvolutional(h=37,w=37,c=256)
batch_normalization_7	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[conv2d_7]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block4_sepconv1	(SeparableConvolution2DLayer)	256,728	W:{1,256,3,3},pW:{728,256,1,1}	[block4_sepconv1_act]	InputTypeConvolutional(h=37,w=37,c=256)	InputTypeConvolutional(h=37,w=37,c=728)
block4_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block4_sepconv1]	InputTypeConvolutional(h=37,w=37,c=728)	InputTypeConvolutional(h=37,w=37,c=728)
block4_sepconv2_act	(ActivationLayer)	-,-	-	[block4_sepconv1_bn]	InputTypeConvolutional(h=37,w=37,c=728)	InputTypeConvolutional(h=37,w=37,c=728)
block4_sepconv2	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block4_sepconv2_act]	InputTypeConvolutional(h=37,w=37,c=728)	InputTypeConvolutional(h=37,w=37,c=728)
block4_sepconv2_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block4_sepconv2]	InputTypeConvolutional(h=37,w=37,c=728)	InputTypeConvolutional(h=37,w=37,c=728)
block4_pool	(SubsamplingLayer)	-,-	-	[block4_sepconv2_bn]	InputTypeConvolutional(h=37,w=37,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
add_31	(ElementWiseVertex)	-,-	-	[block4_pool,batch_normalization_7]		InputTypeConvolutional(h=19,w=19,c=728)
block5_sepconv1_act	(ActivationLayer)	-,-	-	[add_31]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block5_sepconv1	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block5_sepconv1_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block5_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block5_sepconv1]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block5_sepconv2_act	(ActivationLayer)	-,-	-	[block5_sepconv1_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)

block5_sepconv2	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block5_sepconv2_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block5_sepconv2_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block5_sepconv2]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block5_sepconv3_act	(ActivationLayer)	-,-	-	[block5_sepconv2_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block5_sepconv3	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block5_sepconv3_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block5_sepconv3_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block5_sepconv3]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
add_32	(ElementWiseVertex)	-,-	-	[block5_sepconv3_bn,add_31]	-	InputTypeConvolutional(h=19,w=19,c=728)
block6_sepconv1_act	(ActivationLayer)	-,-	-	[add_32]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block6_sepconv1	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block6_sepconv1_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block6_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block6_sepconv1]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block6_sepconv2_act	(ActivationLayer)	-,-	-	[block6_sepconv1_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block6_sepconv2	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block6_sepconv2_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block6_sepconv2_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block6_sepconv2]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block6_sepconv3_act	(ActivationLayer)	-,-	-	[block6_sepconv2_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block6_sepconv3	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block6_sepconv3_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block6_sepconv3_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block6_sepconv3]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
add_33	(ElementWiseVertex)	-,-	-	[block6_sepconv3_bn,add_32]	-	InputTypeConvolutional(h=19,w=19,c=728)
block7_sepconv1_act	(ActivationLayer)	-,-	-	[add_33]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block7_sepconv1	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block7_sepconv1_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block7_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block7_sepconv1]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block7_sepconv2_act	(ActivationLayer)	-,-	-	[block7_sepconv1_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block7_sepconv2	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block7_sepconv2_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block7_sepconv2_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block7_sepconv2]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)

block7_sepconv3_act	(ActivationLayer)	-,-	-	[block7_sepconv2_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block7_sepconv3	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block7_sepconv3_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block7_sepconv3_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block7_sepconv3]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
add_34	(ElementWiseVertex)	-,-	-	[block7_sepconv3_bn,add_33]	-	InputTypeConvolutional(h=19,w=19,c=728)
block8_sepconv1_act	(ActivationLayer)	-,-	-	[add_34]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block8_sepconv1	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block8_sepconv1_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block8_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block8_sepconv1]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block8_sepconv2_act	(ActivationLayer)	-,-	-	[block8_sepconv1_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block8_sepconv2	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block8_sepconv2_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block8_sepconv2_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block8_sepconv2]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block8_sepconv3_act	(ActivationLayer)	-,-	-	[block8_sepconv2_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block8_sepconv3	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block8_sepconv3_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block8_sepconv3_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block8_sepconv3]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
add_35	(ElementWiseVertex)	-,-	-	[block8_sepconv3_bn,add_34]	-	InputTypeConvolutional(h=19,w=19,c=728)
block9_sepconv1_act	(ActivationLayer)	-,-	-	[add_35]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block9_sepconv1	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block9_sepconv1_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block9_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block9_sepconv1]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block9_sepconv2_act	(ActivationLayer)	-,-	-	[block9_sepconv1_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block9_sepconv2	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block9_sepconv2_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block9_sepconv2_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block9_sepconv2]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block9_sepconv3_act	(ActivationLayer)	-,-	-	[block9_sepconv2_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block9_sepconv3	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block9_sepconv3_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)

block9_sepconv3_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block9_sepconv3]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
add_36	(ElementWiseVertex)	-,-	-	[block9_sepconv3_bn,add_35]	-	InputTypeConvolutional(h=19,w=19,c=728)
block10_sepconv1_act	(ActivationLayer)	-,-	-	[add_36]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block10_sepconv1	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block10_sepconv1_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block10_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block10_sepconv1]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block10_sepconv2_act	(ActivationLayer)	-,-	-	[block10_sepconv1_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block10_sepconv2	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block10_sepconv2_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block10_sepconv2_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block10_sepconv2]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block10_sepconv3_act	(ActivationLayer)	-,-	-	[block10_sepconv2_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block10_sepconv3	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block10_sepconv3_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block10_sepconv3_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block10_sepconv3]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
add_37	(ElementWiseVertex)	-,-	-	[block10_sepconv3_bn,add_36]	-	InputTypeConvolutional(h=19,w=19,c=728)
block11_sepconv1_act	(ActivationLayer)	-,-	-	[add_37]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block11_sepconv1	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block11_sepconv1_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block11_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block11_sepconv1]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block11_sepconv2_act	(ActivationLayer)	-,-	-	[block11_sepconv1_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block11_sepconv2	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block11_sepconv2_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block11_sepconv2_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block11_sepconv2]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block11_sepconv3_act	(ActivationLayer)	-,-	-	[block11_sepconv2_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block11_sepconv3	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block11_sepconv3_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block11_sepconv3_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block11_sepconv3]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
add_38	(ElementWiseVertex)	-,-	-	[block11_sepconv3_bn,add_37]	-	InputTypeConvolutional(h=19,w=19,c=728)
block12_sepconv1_act	(ActivationLayer)	-,-	-	[add_38]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)

block12_sepconv1	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block12_sepconv1_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block12_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block12_sepconv1]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block12_sepconv2_act	(ActivationLayer)	-,-	-	[block12_sepconv1_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block12_sepconv2	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block12_sepconv2_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block12_sepconv2_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block12_sepconv2]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block12_sepconv3_act	(ActivationLayer)	-,-	-	[block12_sepconv2_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block12_sepconv3	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block12_sepconv3_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block12_sepconv3_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block12_sepconv3]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
add_39	(ElementWiseVertex)	-,-	-	[block12_sepconv3_bn,add_38]	-	InputTypeConvolutional(h=19,w=19,c=728)
block13_sepconv1_act	(ActivationLayer)	-,-	-	[add_39]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
conv2d_8	(ConvolutionLayer)	728,1024	W:{1024,728,1,1}	[add_39]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=10,w=10,c=1024)
block13_sepconv1	(SeparableConvolution2DLayer)	728,728	W:{1,728,3,3},pW:{728,728,1,1}	[block13_sepconv1_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
batch_normalization_8	(BatchNormalization)	1024,1024	gamma:{1,1024},beta:{1,1024},mean:{1,1024},var:{1,1024}	[conv2d_8]	InputTypeConvolutional(h=10,w=10,c=1024)	InputTypeConvolutional(h=10,w=10,c=1024)
block13_sepconv1_bn	(BatchNormalization)	728,728	gamma:{1,728},beta:{1,728},mean:{1,728},var:{1,728}	[block13_sepconv1]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block13_sepconv2_act	(ActivationLayer)	-,-	-	[block13_sepconv1_bn]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=728)
block13_sepconv2	(SeparableConvolution2DLayer)	728,1024	W:{1,728,3,3},pW:{1024,728,1,1}	[block13_sepconv2_act]	InputTypeConvolutional(h=19,w=19,c=728)	InputTypeConvolutional(h=19,w=19,c=1024)
block13_sepconv2_bn	(BatchNormalization)	1024,1024	gamma:{1,1024},beta:{1,1024},mean:{1,1024},var:{1,1024}	[block13_sepconv2]	InputTypeConvolutional(h=19,w=19,c=1024)	InputTypeConvolutional(h=19,w=19,c=1024)
block13_pool	(SubsamplingLayer)	-,-	-	[block13_sepconv2_bn]	InputTypeConvolutional(h=19,w=19,c=1024)	InputTypeConvolutional(h=10,w=10,c=1024)
add_40	(ElementWiseVertex)	-,-	-	[block13_pool,batch_normalization_8]	InputTypeConvolutional(h=10,w=10,c=1024)	
block14_sepconv1	(SeparableConvolution2DLayer)	1024,1536	W:{1,1024,3,3},pW:{1536,1024,1,1}	[add_40]	InputTypeConvolutional(h=10,w=10,c=1024)	InputTypeConvolutional(h=10,w=10,c=1536)
block14_sepconv1_bn	(BatchNormalization)	1536,1536	gamma:{1,1536},beta:{1,1536},mean:{1,1536},var:{1,1536}	[block14_sepconv1]	InputTypeConvolutional(h=10,w=10,c=1536)	InputTypeConvolutional(h=10,w=10,c=1536)
block14_sepconv1_act	(ActivationLayer)	-,-	-	[block14_sepconv1_bn]	InputTypeConvolutional(h=10,w=10,c=1536)	InputTypeConvolutional(h=10,w=10,c=1536)

block14_sepconv2	(SeparableConvolution2DLayer)	1536,2048	W:{1,1536,3,3},pW:{2048,1536,1,1}	[block14_sepconv1_act]	InputTypeConvolutional(h=10,w=10,c=1536)	InputTypeConvolutional(h=10,w=10,c=2048)
block14_sepconv2_bn	(BatchNormalization)	2048,2048	gamma:{1,2048},beta:{1,2048},mean:{1,2048},var:{1,2048}	[block14_sepconv2]	InputTypeConvolutional(h=10,w=10,c=2048)	InputTypeConvolutional(h=10,w=10,c=2048)
block14_sepconv2_act	(ActivationLayer)	-,-	-	[block14_sepconv2_bn]	InputTypeConvolutional(h=10,w=10,c=2048)	InputTypeConvolutional(h=10,w=10,c=2048)
avg_pool	(GlobalPoolingLayer)	-,-	-	[block14_sepconv2_act]	InputTypeConvolutional(h=10,w=10,c=2048)	InputTypeConvolutional(h=1,w=1,c=2048)
predictions	(DenseLayer)	2048,7	W:{2048,7},b:{1,7}	[avg_pool]	InputTypeConvolutional(h=1,w=1,c=2048)-->InputTypeFeedForward(2048)	InputTypeFeedForward(7)

Příloha 3: Nejvýznamnější publikace

Scopus:

HANZLÍK, P. – KOŽÍŠEK, F. – PAVLÍČEK, J. Design of intelligent decision support systems in agriculture. *International Journal of Mathematics and Computers in Simulation,* 2015, roč. 9, č. , s. 113-118. ISSN: 1998-0159.

MARTÍNEK, T. – HANZLÍK, P. Analysis of the Structure of Job Offers on the Czech Labour Market. *Národohospodářský obzor. (Review of Economic perspectives.) ,* 2014, roč. VOL. 14, č. ISSUE 3, s. 287-306. ISSN: 1213-2446 .

TŮMA, J. – HANZLÍK, P. Automated model transformation method from BORM to BPMN . *Applied Mathematical Sciences,* 2015, roč. 9, č. 116, s. 5769-5777. ISSN: 1312-885X.

GOJDA, O. – HANZLÍK, P. – KLIMEŠOVÁ, D. Study on processing and georeferencing of historical features. *International Journal of Mathematics and Computers in Simulation,* 2015, roč. 9, č. , s. 113-118. ISSN: 1998-0159.

Příspěvky v konferenčních sbornících:

HANZLÍK, P. – PAVLÍČEK, J. – TŮMA, J. Intelligent M-Learning Application for Plant Leaf Recognition. In *Efficiency and Responsibility in Education 2014 05.06.2014, Praha.* : , 2014. s. 179-185.

GOJDA, O. – MARTÍNEK, T. – **HANZLÍK, P**. The Career Perceptions of Future Graduates. In *22nd International Scientific Conference on Agrarian Perspectives - Development Trends in Agribusiness 17.09.2013, CULS, Kamycka 129, Prague 16521 6, Suchdol, CZ.* Prague: CZECH UNIVERSITY LIFE SCIENCES PRAGUE, DEPT SYSTEMS ENG, 2013. s. 125-135.

PAVLÍČEK, J. – ŠVEC, V. – TICHÁ, I. – **HANZLÍK, P**. Business Games Powered by Artificial Intelligence in Education. In *Efficiency and Responsibility in Education 2014 05.06.2014, Prague.* Prague: , 2014. s. 186-195.

VOSTROVSKÝ, V. – TYRYCHTR, J. – **HANZLÍK, P.** Possibilities of Use of the Active Knowledge Databases in the Agricultural Sector. In *Automation Control Theory Perspectives in Intelligent Systems: Proceedings of the 5th Computer Science On-line Conference 2016 (CSOC2016), Vol 3 27.04.2016, online.* Switzerland: Springer International Publishing, 2016. s. 383-390.

Recenzované články:

MARTÍNEK, T. – **HANZLÍK, P.** Analýza vlivu internetových sociálních sítí na výběr nových zaměstnanců v České republice. *ACC Journal,* 2013, roč. Issue B, č. 55-039-13, s. 77-88. ISSN: 1803-9782 .

KEDAJ, P. – PAVLÍČEK, J. – **HANZLÍK, P**. Effective Mind Maps in E-learning. *Acta Informatica Pragensia,* 2015, roč. 3, č. 3, s. 239-250. ISSN: 1805-4951.

TŮMA, J. – PÍCKA, M. – **HANZLÍK, P**. Generated Report of the ORD BORM Model. *Acta Informatica Pragensia,* 2015, roč. 4, č. 1, s. 30-43. ISSN: 1805-4951.