Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Analýza pohybu včelstva po plástu

Bakalářská práce

Jan Humpál

Školitel: Ing. Miroslav Skrbek, Ph.D.

České Budějovice 2018

HUMPÁL, Jan. *Analýza pohybu včelstva po plástu. [Analysis of movement of bees on honeycomb. Bc. Thesis, in Czech.].* České Budějovice, Czech Republic, 2018. Bakalářská práce. Faculty of Science, University of South Bohemia.

Anotace:

This Bachelor's thesis considers the application of methods of computer vision and use of convolutional neural networks for purpose of detection and localization of the queen bee crawling amongst other bees on a honeycomb removed from a beehive. A research of such methods is done together with an attempted implementation, keeping in mind a possibility of using the results of this thesis for implementing a mobile phone application.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě, elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích

dne:.....

.....

Podpis autora práce

Poděkování

Na tomto místě bych rád poděkoval členům svojí rodiny, bez kterých by moje studium nebylo možné.

Moje poděkování patří těm, kteří mi zapůjčili nahrávací techniku.

Speciální poděkování patří také členům ZO ČSV Č. Budějovice za zázemí pro pořízení nahrávek včelstev a panu Jiřímu Ryšavému za prvotní ideu identifikace včelí matky na včelím plástu.

Děkuji Ing. Václavu Krištůfkovi, CSc. za ochotu a pomoc při pořízování těchto záběrů i za odborné vedení v oblasti biologického aspektu této práce.

Veliké díky patří mému školiteli Ing. Miroslavu Skrbkovi, Ph.D. za trpělivé citlivé a odborné vedení při vytváření této práce.

Obsah

1 Úvod

Výpočetní technologie jsou neustále vyvíjeny tak, aby se navyšovaly jejich kapacity a klesaly jejich nároky. Je tedy přirozené, že si nachází své uplatnění ve stále rostoucím množství oblastí lidského působení. Jednou z těchto oblastí, na kterou se zaměřuje i tato práce, je včelařství.

Práce vznikla ve volné návaznosti na projekt Inteligentní včelí úl (IVÚ), který je realizován v rámci projektu Strategie AV 21, Rozmanitost života a zachování ekosystémů na Biologickém centru AV ČR, v. v. i. v Č. Budějovicích. Projekt IVÚ řeší konstrukci včelího úlu osazeného senzory pro snímání mimo jiné například teploty, koncentrace plynů, nebo zvukových projevů včel a následné zpracování sbíraných dat metodami data miningu.

Tato práce se podle provedené rešerše jako první zabývá tematem detekce a lokalizace včelí matky pomocí metod strojového vidění a neuronových sítí. Experimenty v těchto oblastech popsané v této práci, mimo jiné ukazují, že včelí matku lze mezi ostatními včelami s pomocí vhodného modelu detekovat.

1.1 Motivace

Jedno včelstvo může dosahovat velikosti až 50 000 jedinců, kteří se rozdělují na tři kasty. Těmi jsou dělnice, trubci a včelí matka. Zatímco dělnice a trubci dosahují četností řádově destitisíců respektive stovek, matka připadá na každé včelstvo jedna[1].

Práce včelaře zahrnuje úkony, při kterých manipuluje s rámky na nichž se včely pohybují. Při takové manipulaci hrozí nebezpečí umáčknutí včel. V případě dělnic nebo trubců se nejedná o nepřípustnou ztrátu, horší situace nastává, pokud při práci včelař svoje včelstvo takto připraví o matku. Je tedy vhodné matku dočasně separovat odchycením do takzvané klešťové výchytky. Takový odchyt nebo případně pouhá kontrola matky, která je součástí diagnostiky při různých problémových stavech, však nejprve vyžaduje matku v úlu mezi ostatními včelami nalézt a identifikovat. Z tohoto důvodu si někteří včelaři matku značí barevným terčíkem, což, byť to může být považováno za dobrou praxi, není zcela běžné. Navíc i tomuto úkonu předchází nalezení matky neznačené, což může být obtížné nejen pro nezkušeného včelaře. Právě nalezení neznačené matky je úkol, k jehož zjednodušení by tato práce měla svým výsledkem přispět.

Obrázek 1: Hledání neznačené matky.

Čtenář si může udělat představu o situaci, ve které se včelař nachází při hledání neznačené matky na základě obr. 1, na kterém lze pozorovat jeden rámek se včelím dílem a včelami, mezi kterými se nachází matka.[1]

1.2 Cíl práce

Cílem práce je prozkoumat a otestovat metody, které by umožnily strojové zpracování a vyhodnocení snímků rámků se včelami. Takové vyhodnocení by mělo vést k detekci přítomnosti včelí matky a případně její lokalizaci. V případě úspěšné implementace je možné převzít výsledky práce při tvorbě mobilní aplikace, která by včelaři při snímání rámku kamerou telefonu indikovala, zda se v záběru nachází včelí matka a případně vyznačila její polohu.

2 Analýza problému

Na obr. 2 lze pozorovat velmi dobře viditelnou neznačenou včelí matku.[2] Od ostatních včel se zjevně liší svojí velikostí, dosahuje 20-25 mm, ostatní včely dosahují velikosti 15-17 mm[2].

Dále si lze všimnout, že matka má oproti ostatním včelám méně výrazné až neznatelné

[1] Včelí dílo je oboustranné a rámků se v jednom úlu běžně nalézá až 10.
[2] Zhruba uprostřed, ohraničená volným místem.

Obrázek 2: Včelí matka.

rozlišení světlých a tmavých pruhů na zadečku. Nohy matky jsou výrazně světlejší a delší než u ostatních včel, avšak na většině snímků jsou z větší části překryté ostatními včelami. Ze snímku se může zdát, že matka má znatelně tmavší štítek než jiné včely, tento detail se ovšem vytrácí s různým nasvětlením.

V souhrnu se tedy zdá, že markanty včelí matky jsou buďto nespolehlivé, nebo těžce detekovatelné. Naproti tomu ostatní včely se jeví jako dobře rozlišené od pozadí právě výraznými pruhy. Pro postup v rovině strojového vidění za použití klasických algoritmů, což je téma kapitoly 4, byla výchozím bodem následující hypotéza: „Pohyb včelí matky po plástu způsobuje anomálie v pohybu ostatních včel.“ Tuto hypotézu podporuje nejen zběžné pozorování chování včel při pořizování nahrávek, ale i sebrané poznatky z několika studií v knize Včelařská encyklopedie, které hovoří o tzv. mateří látce, kterou matka vylučuje celým povrchem těla. Tuto mateří látku ostatní včely olizují nejen z matky samotné, ale i z míst, kudy matka prošla[2, s. 126, 127].

Na základě této hypotézy je problém lokalizace matky tímto převeden na problém lokalizace včel a následné analýzy jejich pohybu.

3 Použité vybavení

3.1 Nahrávací technika

Pro účely této práce byly pořízeno několik nahrávek včel pohybujících se po plástu spolu se včelí matkou. Do užšího výběru pro další zpracování se dostaly dvě nahrávky, především z toho důvodu, že na ostatních byla včelí matka již značená, což nebylo vhodné pro učení neuronových sítí za účelem vyhledávání matky neznačené.

Prvmí z těchto dvou nahrávek byla pořízena za použití fotoaparátu Sony DSC-HX60. Filosofií této nahrávky byla snaha o napodobení nahrávání koncovým uživatelem, přístroj byl proto nastaven ke snímání v automatickém režimu a byl držen v ruce bez další opory, což dává vzniknout značným translací a změnám v měřítku. Vzniklá nahrávka má rozlišení 1920*1080 při 50 fps.[3] Obrázek 2 je snímkem z této nahrávky.

Druhá nahrávka byla pořízena fotoaparátem Nikon D5600. Při druhém nahrávání bylo cílem pořídit statický obraz pro účely snazšího zpracování, nahrávka tedy vznikala ze stativu. Bylo zachováno rozlišení 1920*1080 při nižším fps 24; což je pro následné zpracování stále dostatečné. Obrázek 1 je snímkem z druhé nahrávky.

3.2 HW pro učení neuronových sítí

Veškeré učení všech testovaných modelů bylo uskutečněno za použití autorova osbního počítače. Sestava zahrnuje procesor AMD Ryzen 5 1600, grafickou kartu Radeon RX 580 a 16 GB paměti RAM. Bližší specifikace uvedeného hardware v tabulkách 1 a 2.

Počet jader / vláken	6 / 12
Základní / boost frekvence	~ 3,2 / 3,6 GHz

Tabulka 1: Procesor

Výpočetní jednotky	36
Maximální výkon	6,2 TFLOP
Paměť	8 GB
Max. šířka pásma paměti	256 GB/s

Tabulka 2: Grafická karta

[3] Snímky za sekundu z anglického *frames per second*.

4 Strojové vidění

Zpracování obrazu pro účely rozpoznávání objektů v něm se podle Hlaváče a Šonky skládá z následujících základních kroků:[3, s. 13]

1. *Snímání, digitalizace a uložení obrazu v počítači.*
2. *Předzpracování.*
3. *Segmentace obrazu na objekty.*
4. *Popis objektů.*
5. *Porozumění obsahu obrazu (často jen klasifikace objektů).*

Samotné pořizování nahrávek je do jisté míry předurčeno dostupnou technikou a je rozebíráno v kapitole 3. Kapitola Strojové vidění se bude zabývat především předzpracováním a segmentací, jelikož popis[4] objektů a jejich klasifikace[5] je pro potřeby této práce triviální respektive irelevantní.

4.1 Zavedení pojmů

V průběhu procesu zpracování obrazu jsou aplikovány různé matematické postupy, je tedy často vhodné o obrazu hovořit v rovině matematického modelu. Pro tuto potřebu bude používán termín **obrazová funkce**, který označuje obecnou funkci $f(i,j) = x_{i,j}$. Obrazová funkce je pro účel popisu digitálního obrazu diskrétní, což je dáno omezeným rozlišením rastrové mřížky snímků, může však být vnímána i jako spojitá při popisu obrazu ve smyslu snímané předlohy. Argumenty funkce reprezentují souřadnice jednotlivých bodů obrazu, také zvaných pixel[6], a hodnota funkce v těchto bodech je nazývána **jas**. Spojité množiny bodů v obraze potom označujeme jako **oblasti**. Množinu bodů, jež mají ortogonálně nebo diagonálně alespoň jeden sousední bod, který oblasti nenáleží, nazýváme **hranice oblasti**. Jako **hrany** ozančujeme oblasti, kde první derivace obrazové funkce nabývá lokálních extrémů, tedy místa, kde v obraze dochází k prudké změně jasu. Hrany jsou užitečné pro segmentaci

[4] Kartézské souřadnice nalezené včely.
[5] Hledány jsou objekty pouze jedné třídy – včela.
[6] Z anglického *picture element*.

tam, kde koincidují s hranicemi oblastí jež jsou předmětem zájmu[7] v obraze. Za nežádoucí pokládáme hrany typicky velikosti několika málo bodů, které vznikly chybou při snímání, mluvíme o **šumu**[3, 4].

Dalším užitečným termínem je **histogram,** jedná se o diskrétní funkci, která pro všechny možné hodnoty jasu daného obrazu udává jejich četnost[4].

4.2 Předzpracování

Při předzpracování je cílem připravit obraz tak, aby byl usnadněn další postup. Předzpracováním se nezískává v obraze žádná nová informace, pouze je některá informace potlačena a jiná zvýrazněna. Cílem předzpracování pro účely této práce je především potlačení šumu, čehož je možno dosáhnout vyhlazováním za použití vhodného filtru. Filtrem se rozumí matice vah h, též nazývána kernel, okénko nebo konvoluční maska[4], která je použita k výpočtu nových hodnot jasu obrazové funkce. Hodnota jasu nového obrazu $f(i,j)$ v každém bodě vznikne jako lineární kombinace lokálního okolí odpovídajícího bodu původního obrazu $g(i,j)$, popsáno vztahem 1[3].

$$f(i,j) = \sum_{m=-\lfloor M/2 \rfloor}^{\lfloor M/2 \rfloor} \sum_{n=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} g(i+m, j+n)h(m,n) \tag{1}$$

Rozměry kernelu M a N jsou voleny tak, aby byly menší, než nejmenší relevantní detail v obraze; používají se většinou lichá čísla, aby měl kernel střed, který pak může korespondovat se zpracovávaným bodem[3]. Tomu je uzpůsobeno i indexování ve výrazu 1. Jako příklad kernelu je možno uvést vztah 2, což je jednoduchý filtr pro vyhlazování průměrováním.

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

Kromě výše popsaných lineárních filtrů existují i nelineární filtry, které definují nové hodnoty obrazové funkce na základě statistických vlastností okolí bodů původního obrazu. Příkladem může být medianový filtr.

[7] Používá se zkratka ROI z anglického *region of interest*.

4.3 Segmentace

Úkolem segemntace je rozdělení obrazu na části, které odpovídají objektům v obraze zachyceným za účelem vyčlenění zájmových oblastí od zbytku obrazu. V závislosti na míře úspěchu je možno hovořit o částečné, nebo úplné segmentaci[3]. *Segmentační algoritmy zpravidla zakládají na jedná ze dvou základních vlastností hodnot jasu, kterými jsou diskontinuity a podobnosti*[4, s. 568, přeloženo]. Diskontinuity v obraze jsme již dříve definovali jako hrany a právě těmi se zabývá detekce hran. S podobnostmi v obraze zase pracují prahovací algoritmy. V dalších sekcích tedy budou rozebírány dvě kategorie segmentačních algoritmů.

4.3.1 Prahování

Segmentační metoda prahování vyniká jednoduchosti své implementace, výpočetní nenáročností a intuitivními vlastnostmi a je zároveň nejstarší metodou segmentace[3, 4]. Ve své nejjednodušší podobě transformuje prahování vstupní obrazovou funkci podle vztahu 3[3], kde T je předem určená hodnota prahu.

$$f(i,j) = \begin{cases} 1 & g(i,j) \geq T \\ 0 & g(i,j) < T \end{cases} \tag{3}$$

Výstupem prahování je tedy binární obraz oblastí, v nichž hodnota jasu v původním obraze překračuje daný práh. Takto je možno v obraze segmentovat objekty se společnou vlastností vysoké odrazivosti. Přímočarou modifikaci prahování popisuje vztah 4[3], kde D je množina úrovní jasu očekávaných u hledaných objektů.

$$f(i,j) = \begin{cases} 1 & g(i,j) \in D \\ 0 & jinak \end{cases} \tag{4}$$

Je zřejmé, že prahování není vhodné ve všech situacích a je silně závislé na volbě hodnoty prahu. Jako příklad metody nalezení správného prahu lze uvést analýzu histogramu obrazu. Pokud je daný obraz dobře segmentovatelný prahování, má jeho histogram bimodální charakter, kdy jeden vrchol reprezentuje hledané objekty a druhý pozadí. Vhodnou hodnotou prahu je potom stupeň jasu, který odpovídá minimu v intervalu mezi dvěma vrcholy. V komplikovanějším případě, kdy je histogram multimodální lze metodu upravit podle vztahu 4

tak, aby množina D postihovala vícero intervalů stupňů jasu v závislosti na rozložení vrcholů histogramu[3].

Na výsledky prahování má zásadní vliv také osvětlení. V případě nerovnoměrného osvětlení nelze dosáhnout úplné segmentace za použití globální hodnoty prahu. Namísto jednotného prahu lze aplikovat adaptivní prahování, kdy je práh určován dynamicky na základě lokálních vlastností obrazové funkce[4].

4.3.2 Detekce hran

Standardem pro detekci hran je Cannyho algoritmus[6], který lze implementovat pomocí následujících kroků:[7]

1. Filtrace šumu pomocí masky aproximující Gaussovu funkci.
2. Výpočet síly hran z vertikálního a horizontálního gradientu.
3. Výpočet směru hran.
4. Označení hran podle jejich směru.
5. Ztenčení hran potlačením hodnot, které nejsou lokálními maximy.[8]
6. Finalizace hran hysterezí s dvěma prahy.

K diskrétní aproximaci vertikálního a horizontálního gradientu v obraze je možno použít Sobelův operátor[8], což je dvojice filtrů popsaná vztahem 5.

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad h_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{5}$$

Následně je síla hran aproximována vztahem 6, kde G_x a G_y je výstup filtrování h_x respektive h_y.

$$G = |G_x| + |G_y| \tag{6}$$

[8] *Non-maximum suppression.*

Směr hran lze určit podle vztahu 7, při ošetření případu, kdy je $G_x = 0$, potom je potřeba se řídit vztahem 8.

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{7}$$

$$\theta = \begin{cases} 0 & G_y = 0 \\ \frac{\pi}{2} & G_y \neq 0 \end{cases} \tag{8}$$

Jelikož v rastrové mřížce lze v bezprostředním okolí bodu rozeznat pouze osm směrů, je na základě vypočítaného úhlu přiřazena hraně orientace odpovídající jedné z vertikální, horizontální a dvou diagonálních os.

Následně jsou potlačeny všechny body, kde síla hrany není větší než síla hrany obou přilehlých bodů sousedících kolmo k přiřazené orientaci.[9]

Na závěr, s pomocí dvou předem určených hodnot prahů T_1 a T_2, $T_1 < T_2$, jsou nejprve potlačeny body se silou hrany nižší než T_1 a zachovány body se silou hrany vyšší než T_2. Poté jsou nerozhodnuté body z intervalu $\langle T_1, T_2 \rangle$ zachovány, pokud přímo sousedí s jiným původně zachovaným bodem, nebo jsou s takovým bodem spojeny dalšími nerozhodnými body.

4.3.3 Houghova transformace

Metoda detekce přímek v obraze založená na patentu Paula Hougha[9], později zdokonalená Dudou a Hartem[10], pracuje s globálními vztahy pixelů[4], s čímž souvisí i její značná odolnost vůči šumu[3].

Uvažujme parametrickou rovnici přímky, danou vztahem 9, kde ρ je vzdálenost přímky od počátku souřadnicové soustavy a θ je úhel, který svírá normála udávané přímky s osou x.

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{9}$$

Houghova transformace pracuje s akumulátorem, což je matice, která ukládá hlasy pro

[9] Tedy ve směru a proti směru gradientu.

množinu prohledávaných přímek. Jednotlivé rozměry matice odpovídají kvantizaci parametrů ρ a θ, takže každé jedno pole matice reprezentuje jednu možnou přímku v obraze a velikost matice koresponduje s přesností metody. Za použití akumulátoru algoritmus prochází jednotlivé pixely a použitím vhodného rozhodovacího kritéria určí, zda pixel náleží přímkám popsaným akumulátorem, které jím mohou procházet. Přímkám, kterým pixel náleží je připočten hlas. Po zpracování celého obrazu určují maxima akumulátoru pravděpodobnou polohu přímek[5].

Houghovu transformaci lze zobecnit pro libovolnou křivku navýšením dimenzionality akumulátoru adekvátně počtu parametrů dané křivky[3].

5 Neuronové sítě

Tato oblast strojového učení zaznamenala v posledních letech značný nárust v zájmu v akademických řadách[11]. Algoritmy strojového učení fungují na principu zpracování dat, ze kterých jsou vyvozovány pravidelnosti. Tento druh samostatně nabytých znalostí ve formě datového modelu může pak počítač využít k řešení problémů, jež zdánlivě vyžadují pochopení reálného světa na subjektivní úrovni, například v oblastech diagnostické medicíny, tržní analýzy, zpracování přirozeného jazyka a rozpoznávání obrazu[12, 13]. Neuronové sítě, koncept původně inspirovaný stavbou lidského mozku, můžeme nahlížet jako výpočetní systém, jenž strojové učení zprostředkovává pomocí série učených funkcí, které transformují vstupní data na požadovaný výstup, typicky jiného charakteru[11].

5.1 Zavedení pojmů

Základní stavební jednotkou sítě je **neuron**, který transformuje prvky vstupního vektoru x na výstupní hodnotu. K tomuto výpočtu slouží **váhy** w,[10] podle nichž jsou různě zohledňovány jednotlivé složky vstupního vektoru a **práh** b.[11] Výstup neuronu y je pak určen nelinární **aktivační funkcí** podle vztahu 10[15].

$$y = f\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{10}$$

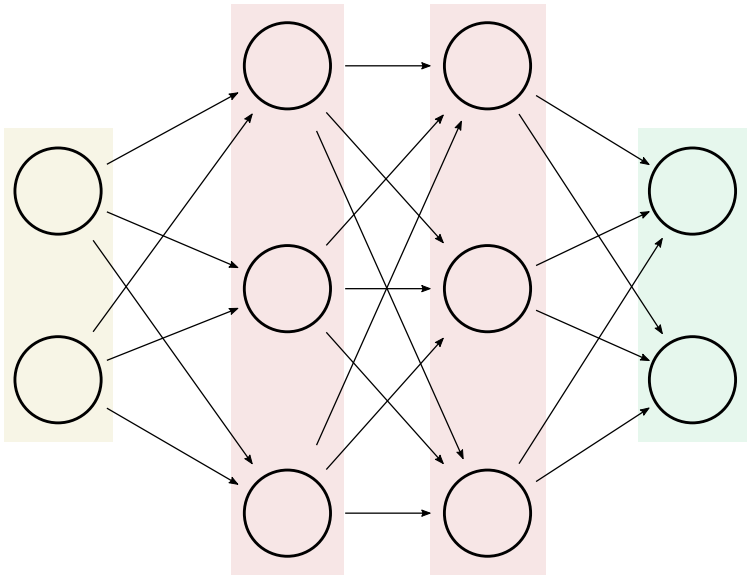
[10] Z anglického *weight*.
[11] Z anglického *bias*.

Příkladem aktivační funkce může být logistická funkce definovaná vztahem 11, nebo tzv. ReLU[12] funkce definovaná vztahem 12[15].

$$f(x) = \frac{1}{1 + e^{-\gamma x}} \tag{11}$$

$$f(x) = \max(0, x) \tag{12}$$

Neurony jsou sdružovány do **vrstev**, které dalším skládáním podle zvolené architektury tvoří neuronovou síť. Speciálně mluvíme o vstupní vrstvě, jejíž tvar je určen charakterem vstupních dat a výstupní vrstvě, jejíž tvar je dán požadavkem na výstup sítě. Vrstvy mezi vstupem a výstupem označujeme jako **skryté**. Vrstvy jsou mezi sebou běžně propojeny tak, že výstupy vrstvy i tvoří vstupy následující vrstvy $i + 1$[14]. Jako ilustrační příklad malé neuronové sítě je uveden obrázek 3.



Obrázek 3: Schéma jednoduché plně propojené neuronové sítě se dvěma skrytými vrstvami (červeně).

Tato práce se zabývá sítěmi učenými pomocí tzv. *supervised learning*, tedy **učení s učitelem**, při kterém jsou síti předkládány vzory z **trénovací množiny**, které reprezentují data, jež se má síť naučit zpracovávat. Odezva sítě je pak porovnávána s očekávaným výsledkem, na základě čehož jsou upravovány váhy a prahy tak, aby se výstup sítě očeká-

[12] Z anglického *rectified linear unit*.

vanému výsledku přiblížil. K tomuto procesu dochází opakovaně, kdy jedné iteraci se říká **epocha**. Pro vyhodnocení schopnosti sítě reagovat na neučená data se používá tzv. **testovací množina**. Pokud se učením zlepšuje odezva sítě na vzory z trénovací množiny, ale odezva na testovací množinu zůstává stejná, nebo se zhoršuje, dochází k **přeučení** a síť ztrácí schopnost **generalizace**[13].

Pro algoritmus učení je nezbytné zavést kritérium hodnotící výstup neuronové sítě, **chybovou funkci**, příkladem může být vztah 13, kde N je počet vzorů v trénovací množině, y je odezva sítě a $\hat{y}$ je očekávaná odezva[14, 13].

$$E = \frac{1}{n} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{13}$$

Proces učení je potom optimalizační úloha hledání minima chybové funkce. Známým řešením takové úlohy je sestup podle gradientu,[13] kdy se s pomocí první derivace funkce hledá taková úprava vstupního parametru, která generuje žádoucí změnu výstupu. Jelikož chybová funkce je závislá na mnoha parametrech, používá se výpočet gradientu $\nabla_x f(x)$, tedy vektoru parciálních derivací chybové funkce podle jednotlivých parametrů, podle kterého jsou nové parametry určeny vztahem 14, kde t označuje index epochy a parametr ϵ se nazývá **rychlost učení**[13, 14].

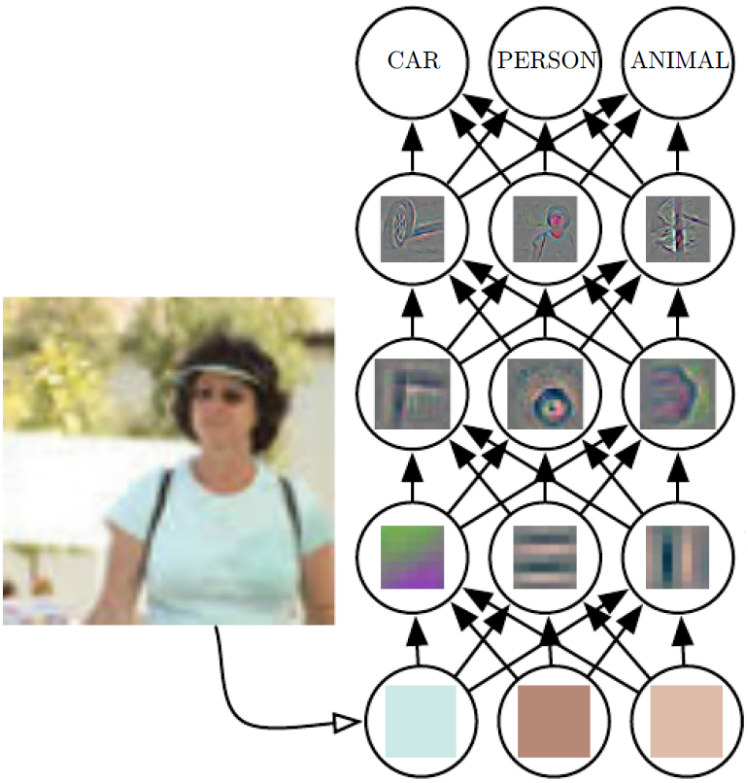
$$w_{t+1} = w_t - \epsilon \nabla_x f(x) \tag{14}$$

Pro úpravu vah celé sítě šířením informace získané chybovou funkcí od výstupních vrstev sítě směrem ke vstupním vrstvám se používá algoritmus **Backpropagation**[13].

5.2 Hluboké sítě

Hluboké učení zakládá na principu kombinování jednodušších konceptů pro vyjádření těch složitějších, čímž umožňuje počítači složité koncepty zpracovat. Na datovém modelu to můžeme chápat jako hloubku výpočetního grafu, nebo hloubku grafu, který popisuje hierarchickou návaznost jednotlivých konceptů. U neuronových sítí můžeme v tomto smyslu pozorovat početjejich skrytých vrstev, kde každá z nich reprezentuje určitou úroveň složitosti, jak je

[13] *Gradient descent*



Obrázek 4: Ilustrace skládání příznaků v obraze neuronovou sítí. Obrázek na vstupu je klasifikován výstupem. Převzato z [13].

ilustrováno na obrázku 4 [13].

Neexistuje však jednoznačný konsenzus, který by udával přesnou hranici která rozlišuje hluboké učení od mělkého, hovoříme-li tedy o hlubokém učení, máme na mysli modely, které vykazují větší míru skládání konceptů, než je obvyklé u klasických metod strojového učení[13].

V dalších sekcích jsou popsané některé úspěšné architektury hlubokých neuronových sítí.

5.2.1 LeNet

LeNet je průkopnická neuronová síť navržená Yann LeCunem et al., která byla narozdíl od svých předchůdců učena přímo na rozpoznávaných obrázcích, čímž ukazuje zbytnost předzpracování obrazových dat pro účely strojového učení. Namísto předem vytvořených filtrů je síť ponechána aby se naučila filtry vlastní, které jsou tak šité na míru pro řešení dané úlohy[16].

K tomu účelu slouží konvoluční vrstvy, které transformují trojrozměrný vstup[14] filtrováním sadou kernelů na trojrozměrný výstup[15] nazývaný mapa příznaků.[16] Konvoluční vrstvy mají tři zásadní výhody, kterými jsou řídké propojení, sdílení parametrů a ekvivariance vůči posunutí. První dvě výhody přispívají ke značné redukci množství parametrů sítě a tím i k redukci paměťových a časově výpočetních nároků modelu. Ekvivariance vůči posunutí zajišťuje možnost detekovat objekty v obraze bez ohledu na jejich pozici[13].

5.2.2 AlexNet

Alex Krizhevsky et al. se svou konvoluční sítí vyhrál v roce 2012 soutěž ILSVRC, která plní funkci benchmarku v oblasti klasifikace objektů v obraze[18]. AlexNet ve své době představil řadu inovací v oblasti hlubokého učení, jimiž jsou mj. ReLU aktivační funkce, pooling s překrýváním, dropout a trénování s použitím dvou GPU[17].

Přínos ReLU aktivace spočívá ve velkém a konzistentním gradientu v části, kde je jednotka aktivní, což značně urychluje učení[13, 17].

Pooling je bezparametrická vrstva nasazovaná za aktivací konvoluční vrstvy, která redukuje velikost výstupu nahrazováním čtvercových oblastí určité velikosti jejich statistickou sumarizací, typickým příkladem je maxpooling, který vybírá z každé oblasti maximum. Poolingu s sebou přináší v závislosti na implementaci různé výhody, především zrychlení procesu učení redukcí počtu parametrů sítě a určitou míru invariance vůči posunutí[13]. Pooling s překryvem v síti Alexnet napomáhá zabránit přeučení[17].

Termín dropout označuje techniku, kdy je s určitou pravděpodobností výstup každého neuronu v ovlivněné skryté vrstvě nastaven na nulu. Při učení Alexnetu byl použit dropout s pravděpodobností 0,5 ve dvou plně propojených vrstvách, což údajně prodloužilo trénovací čas dvojnásobně, ale zároveň značně redukovalo přeučení[17].

Krizhevsky et al. ve své publikaci popisuje potřebu paralelizace architektury pro dvě GPU z důvodu paměťových nároků trénovaného datasetu a zároveň na základě výsledků predikuje, že další přímočaré zlepšení je podmíněno pouze použitím výkonějšího hardwaru.[17]

[14] Na vstupu sítě se jedná o šířku, výšku a barevné kanály obrazu.
[15] Třetí rozměr výstupu odpovídá počtu použitých kernelů.
[16] Z anglického *feature map*.

5.2.3 VGG

Karen Simonyan a Andrew Zisserman, členové Oxfordské Visual Geometry Group ve svém článku z roku 2015 představují sérii experimentů v oblasti konvolučních sítí, kterými ukazují význam hloubky sítě[19].

Za tímto účelem je architektura všech testovaných sítí složená téměř výhradně skládáním jednoho druhu bloku vrstev, který se skládá z jedné až čtyř stejných konvolučních vrstev s kernelem rozměrů 3×3 s ReLU aktivací, následovaných jednou maxpool vrstvou bez překryvu s kernelem 2×2. Po pěti takových blocích následují tři plně propojené vrstvy, což generuje sítě s hloubkou až 19 skrytých parametrických vrstev [19].

Nahrazení jedné konvoluční vrstvy s velkým kernelem serií konsekutivních konvolučních vrstev přináší výhodu redukce počtu parametrů sítě a zároveň nahrazuje jednu aktivaci třemi, což zlepšuje rozhodovací funkci[19].

Architektura VGG se ukázala jako úspěšná, když zaujala 2. místo v soutěži ILSVRC 2014[19, 18].

5.2.4 Inception, GoogLeNet

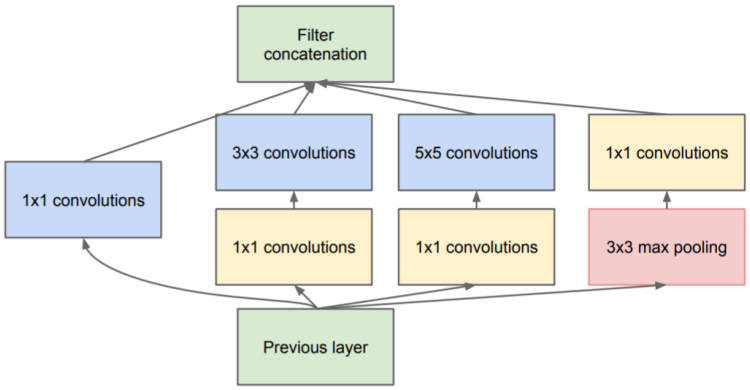
První místo v ILSVRC v roce 2014 obsadil Szegedy et al. se svým příspěvkem GoogLeNet postaveným na architektuře Inception[18, 20].

Hlavní myšlenkou architektury bylo umožnit prohloubení sítě za zachování výpočetních nároků, čehož je dosaženo použitím speciálních konvolučních vrstev s kernelem velikosti 1×1, které na výstupu prakticky produkují lineární kombinaci „kanálů“ svého vstupu. Slouží tak nejen jako další vrstva s aktivační funkcí, ale především jako dimenzionální redukce[20].

Název Inception je inspirován principem hlavního stavebního bloku svojí architektruy, sítě v síti[21], kdy je vstup zpracován paralelně čtyřmi různými vrstvami a výstupy jsou konkatenací připraveny pro další blok, viz obrázek 5.

5.3 Transfer learning

Bylo pozorováno, že hluboké sítě trénované na rozpoznávání objektů v obrazech z reálných prostředí vykazují značnou míru podobnosti filtrů v prvních vrstvách[22]. Transfer learning je metoda, kdy se síť trénovaná pro řešení jednoho druhu úkolu přetrénuje pro řešení úkolu



Obrázek 5: Schéma Inception bloku, převzato z [20].

příbuzného druhu, přičemž se mohou zachovat nezměněny váhy v nižších vrstvách, u kterých se očekává, že budou dobře aplikovatelné i pro nový úkol [13, 22].

6 Experimenty

Byly provedeny experimenty z oblasti strojového vidění a neuronových sítí. Materiál, který byl nasnímaný, jak je popsáno v kapitole 3, byl pro další zpracován rozložen na snímky za použití programu ffmpeg. Pro účely neuronových sítí byla včelí matka na vybraných snímcích označena za použití programu LabelImg.

Pro veškeré programování byl použit programovací jazyk python, který byl vybrán pro vysokou dostupnost knihoven pro analýzu dat a zpracování obrazu. Jako skriptovací jazyk je python i vhodný pro experimentální programování.

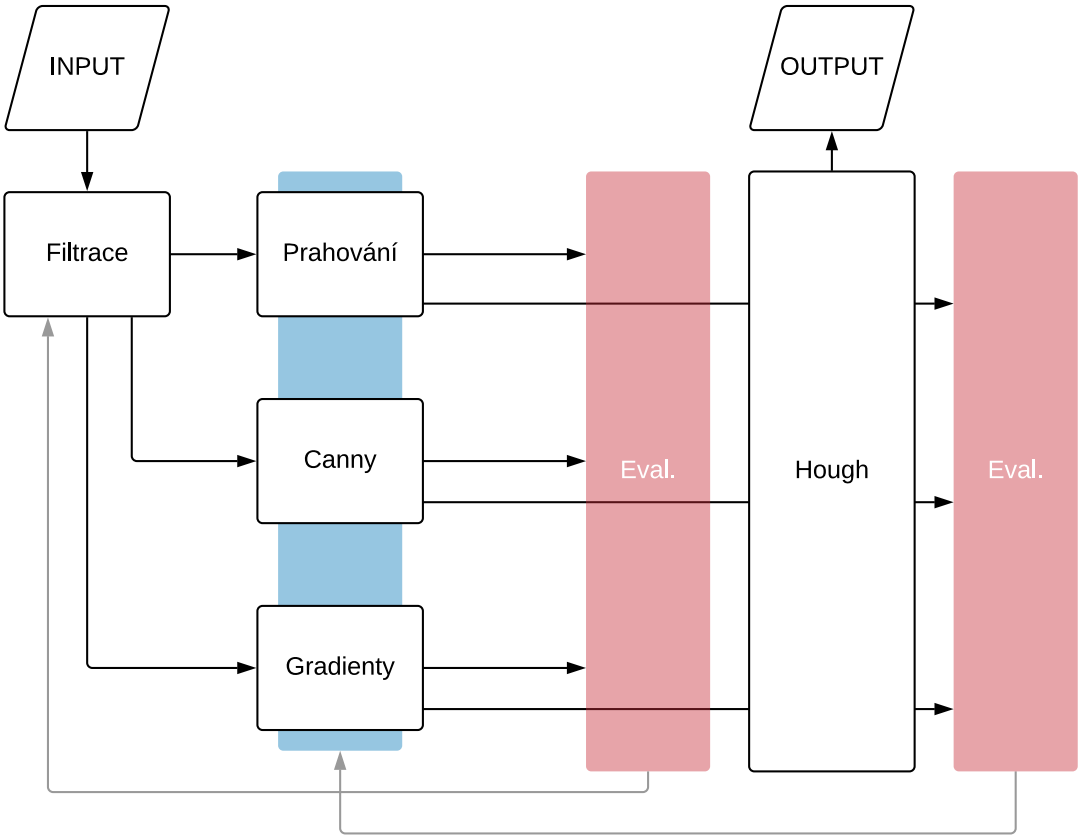
Pro zkoumání klasických metod strojového vidění byla použita knihovna OpenCV (dále jen OCV), která disponuje vhodnými základními funkcemi pro úpravu obrazu.

Pro vývoj neuronových sítí bylo použito API Keras, kde byl jako backend nainstalován framework PlaidML. Ten byl vybrán z důvodu kompatibility s grafickou kartou Radeon.

6.1 Klasické metody strojového vidění

Cílem experimentů v této oblasti bylo nejprve prozkoumat metody zpracování obrazu a aplikovat je tak, aby se za pomocí Houghovy transformace daly detekovat proužky na zadečku včel.

V první etapě práce byly provozovány nesystematické experimenty s funkcemi OCV,



Obrázek 6: Schéma experimentů se strojovým viděním. Modrý blok značí segmentační metody. Červené bloky značí evaluaci výstupů a následnou zpětnou vazbu.

které sice nevedly k žádným užitečným výsledkům, avšak dobře posloužily pro osvojení používání těchto funkcí a pochopení vlivu jejich parametrů na jejich výstup. Na základě těchto experimentů byly vybráno několik funkcí pro filtraci a segmentaci, které byly dále využity. Pro další testování aplikace těchto funkcí a jejich vhodné parametrizace byl použit proces ilustrovaný schematem na obrázku 6.

Jak schema naznačuje, proces probíhá v několika iteracích, kdy jsou parametry jednotlivých bloků nastaveny nejprve arbitrárně a následně laděny s pomocí zpětné vazby poskytnuté výstupem následujícího bloku. Podrobnější popis procesu poskytuje následující seznam jednotlivých kroků.

1. Párování funkcí filtrace a segmentace:
    (a) Filtrace vstupního obrazu vybranými funkcemi s arbitrárním nastavením jejich parametrů.
    (b) Zpracování výstupu kroku 1a segmentačními funkcemi, taktéž s arbitrární parametrizací.
    (c) Vyhodnocení výstupu kroku 1b.
    (d) Výběr párování konkrétních filtračních funkcí pro konkrétních segmentační funkce na základě kroku 1c.
2. Nastavení parametrů funkcí filtrace:
    (a) Generování výstupu filtrace s rozsahem hodnot parametrů.
    (b) Zpracování výstupu kroku 2a segmentačními funkcemi s arbitrárním nastavením parametrů.
    (c) Vyhodnocení výstupu kroku 2b.
    (d) Nastavení parametrů filtrace na základě kroku 2c.
3. Nastavení parametrů funkcí segmentace:
    (a) Generování výstupu segmentace s rozsahem hodnot parametrů.
    (b) Zpracování výstupu kroku 3a Houghovou transformací s arbitrárním nastavením parametrů.
    (c) Vyhodnocení výstupu kroku 3b.
    (d) Nastavení parametrů funkcí segmentace na základě kroku 3c
4. Vyhodnocení výsledků Houghovy transformace.
    (a) Generování výstupu Houghovy transformace s rozsahem hodnot.
    (b) Vyhodnocení výstupu kroku 4a.

Vyhodnocování výstupů bylo prováděno subjektivně na základě dvou faktorů. Prvním a prioritním z nich bylo množství a kvalita zobrazení pruhů na včelím zadečku. Druhým faktorem byla míra šumu ve formě prvků v obraze nežádoucích pro další zpracování.

Blok nazvaný Gradienty je vlastní implementace inspirovaná Cannyho detekcí hran. Jak název napovídá, prvním krokem je výpočet vertikálního a horizontálního gradientu v obraze za pomocí Sobelova operátoru. Následně je vypočítán úhel gradientu, jako je popsáno v sekci 4.3.2. Hodnoty úhlů jsou pak vyhodnoceny posuvným okénkem ve kterém se zpracovává jejich histogram. Analýza histogramu pracuje s hlasovací maticí velikosti původního obrazu, kde je každému bodu zaznamenán hlas vždy, když je hodnota úhlu jeho gradinetu modem v okolí zpracovávaném okénkem. Výstupem je potom binární obraz, kde nenulové hodnoty korespondují s body, kterým byl zaznamenán počet hlasů překračující předem určený práh.

Takto navržená funkce vychází z předpokladu, že v oblastech obrazu, kde se nachází včelí zadeček bude histogram úhlů gradientu unimodální, což způsobí, že body, které neodpovídají hranám mezi pruhy na včelách budou potlačeny nedostatkem počtu hlasů.

Navzdory snaze o optimalizaci maximalizací využití knihovních funkcí, viz obrázek 7 s ukázkou hlavní části implementace, vykazuje funkce značnou časovou náročnost, kdy při větších hodnotách velikosti posuvného okénka a malých hodnotách velikosti kroku okénka trvá výpočet s použitým hardware i více než 10 sekund. Z tohoto důvodu byla použita i alternativní implementace, která vynechává hlasování a pouze každý pixel buďto ponechá, nebo potlačí, podle toho, zda za připuštění parametrizované odchylky je, respektive není modem ve svém lokálním okolí.

```
maxVotes = np.zeros(mat_phases.shape, np.long)
votes = np.zeros(mat_phases.shape, np.uint8)

### Slide a window over the image and process one ROI at a time
for i in range(0, rows + 1, stepSize):
    for j in range(0, cols + 1, stepSize):
        ### Define ROI
        minr, minc = max(i - maskSize//2, 0), max(j - maskSize//2, 0)
        maxr, maxc = min(i + maskSize//2, rows), min(j + maskSize//2, cols)
        roi = mat_phases[minr : maxr, minc : maxc]

        ### Calculate histogram in roi
        hist = cv2.calcHist([roi],[0],None,[128],[0,128])

        ### Find peak in histogram and compute acceptable values
        modus = np.argmax(hist)
        valRange = np.arange(modus - maxModDeviation, modus + maxModDeviation + 1)
        valRange %= 128

        ### Increase vote values
        maxVotes[minr : maxr, minc : maxc] += 1
        indices = np.where(np.isin(roi, valRange))
        shiftIndices = (indices[0] + minr, indices[1] + minc)
        votes[shiftIndices] += 1

### Count votes
maskFinal = np.array(np.where(votes > maxVotes * minVotePercent, 255, 0), dtype=np.uint8)
```

Obrázek 7: Ukázka kódu funkce implementované v bloku „Gradienty“.

V ostatních procesech byly uplatněny funkce popsané tabulkou 3, jejichž testované parametry jsou k vidění v tabulce 4.

Funkce HoughLinesP je implementace, která namísto přímek hledá úsečky pravděpodobnostní metodou.

Proces	Použité funkce knihovny OCV
Filtrace	GaussianBlur, medianBlur, bilateralFilter
Prahování	threshold, adaptiveThreshold
Canny	Canny
Hough	HoughLinesP

Tabulka 3: Použité funkce

Funkce	Parametry	Popis parametrů
GaussianBlur	ksize	velikost kernelu
medianBlur	ksize	velikost kernelu
bilateralFilter	d	velikost okolí zpracovávaného bodu
	sigmaColor	maximální barevná vzdálenost bodů v okolí, které mohou ovlivnit zpracovávaný bod
threshold	thresh	hodnota prahu
adaptiveThreshold	c	posun funkcí nalezeného prahu
	adaptiveMethod	příznak určující druh adaptivního prahování[17]
Canny	Threshold1	spodní práh hystereze
	Threshold2	horní práh hystereze
HoughLinesP	threshold	minimum hlasů pro zachování úsečky
	minLineLength	minimální délka pro zachování úsečky
	maxLineGap	tolerovaná mezera v nalezených úsečkách

Tabulka 4: Testované parametry používaných funkcí.

Vzhledem k časovým důvodům a vzhledem k výstupům, které byly popsaným procesem generovány a které nepůsobily nijak perspektivně pro další zpracování, bylo experimentů v oblasti klasických metod strojového vidění zanecháno ve prospěch experimentů s trénováním neuronových sítí.

6.2 Neuronové sítě

Manuálním labelováním snímků získaných z nahraných materiálů vzniknul dataset pro trénování lokalizace, popsaný tabulkou 5. Labely vzorů v tomto datasetu jsou souřadnice obdélníkového rámečku ohraničujícího v obraze včelí matku. Obrázky byly downsamplovány na rozlišení 480*270.

V oblasti neuronových sítí byl nejprve podniknut experiment s transfer learningem v duchu použití osvědčené architektury pro dosažení dobrého výsledku. Pro tento experiment

[17] Použity byly gausovské a medianové.

Množina	Počet vzorů
Trénovací	722
Testovací	207
Validační	29

Tabulka 5: Dataset pro učení lokalizace včelí matky.

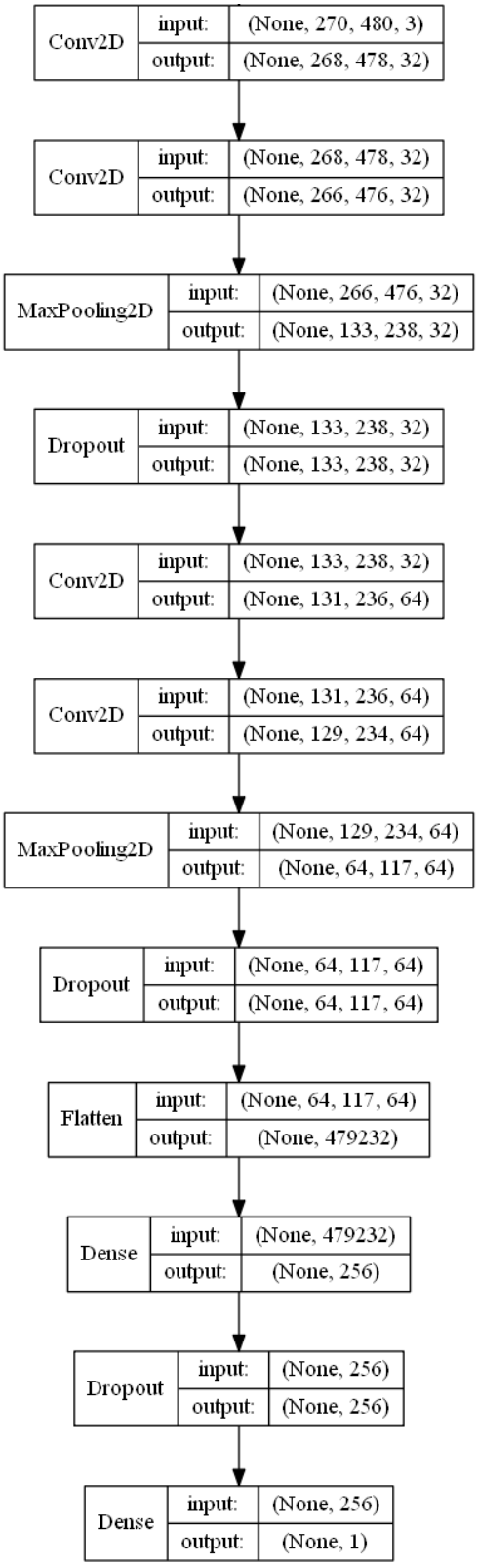
Množina	Počet vzorů
Trénovací	460
Testovací	132
Validační	46

Tabulka 6: Dataset pro učení detekce včelí matky.

byl použit převzatý projekt [23], který využívá frameworku Tensorflow ke trénování předtrénovaného modelu architektury Inception[20].

Další experimenty byly prováděny s vlastními architekturami, které byly inspirovány architekturou VGG[19]. Hlavním stavebním prvkem těchto architektur byl blok skládající se z několika konvolučních vrstev s velikostí kernelu 3, maxpool vrstvy bez překryvu s velikostí kernelu 2 a dropout vrstvy s mírou výpadku 0,25. Předposlední vrstvy byly plně propojené a následované dropout vrstvou s mírou výpadku 0,5. Až na výjimky byla použita aktivace ReLU. Sítě byly učeny na normalizovaných vstupech, algoritmem klesání podle gradientu s rychlostí učení 0,01 po 200 epoch. Jako chybová funkce byla použita střední kvadratická odchylka. Každých 5 epoch bylo aktuální řešení uloženo, při poklesu testovací chyby proti předchozímu nejlepšímu řešení.

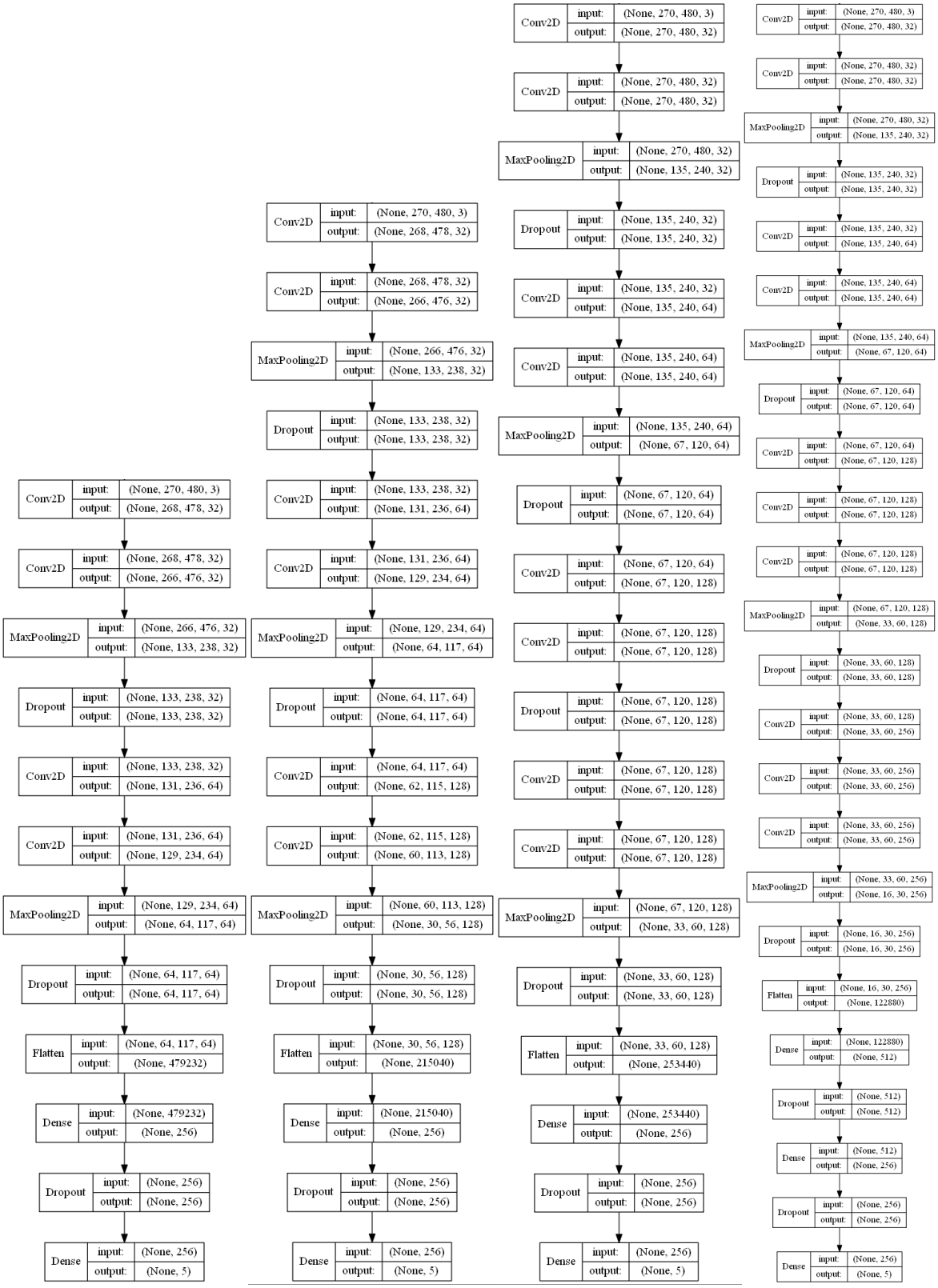
Jako první byl uskutečněn experiment s neuronovou sítí určené pouze k detekci. Účelem bylo ověřit, zda je za použití architektury typu VGG vůbec možné včelí matku v obraze detekovat. Pro tuto síť byl vytvořen speciální dataset popsaný tabulkou 6. Síť má jediný výstup, který udává pravděpodobnost výskytu včelí matky. Jako aktivační funkce výstupní vrstvy byla použita logistická funkce. Samotná architektura je popsána schematem na obrázku 8.



Obrázek 8: Architektura sítě pro detekci.

Po ověření možnosti včelí matku detekovat bylo přistoupeno k trénování lokalizačních sítí. Celkem bylo experimentováno s pěti různými architekturami, jejichž schemata jsou na obrázku 9. Výstupem každé z lokalizačních sítí je vektor pěti hodnot: pravděpodobnost výskytu včelí matky, souřadnice středu její obdelníkové oblasti, šířka a výška obdelníku. Lokalizační hodnoty jsou udávány relativně jako procento rozměrů obrázku, tedy v rozmezí $< 0, 1 >$. Z toho důvodu byla první síť trénována na výstupech s logistickou aktivační funkcí, jelikož její obor hodnot odpovídá oboru hodnot výstupů. Druhá síť s totožnou architekturou a závěrečnou aktivací ReLU sice dosáhla horšího výsledku, avšak při prvotním vyhodnocování došlo k záměně, takže zbylé sítě byly trénovány s aktivací ReLU na konci.

Zbylé tři architektury jsou experimentem v prohlubování sítě, kdy jsou střídavě přidávány bloky a vrstvy v rámci bloků. Za povšimnutí stojí, že čtvrtá architektura zavádí tzv. „same padding“ konvolučních vrstev, který zabraňuje redukování šířky a výšky jejich výstupu tím, že před filtrací orámuje vstupní obraz tak, aby kernel mohl zpracovávat i okrajové pixely. Bez této výplňky je výstup redukován o $\lfloor k/2 \rfloor$ pixelů po všech okrajích, kde k je velikost kernelu. Tato změna byla zavedena, aby se zachovala co největší šířka a výška výstupu posledního bloku před první plně propojenou vrstvou a síť si tak zachovala větší rozlišovací schopnost.



Obrázek 9: Lokalizační architektury. První zleva reprezentuje první dvě sítě.

7 Výsledky

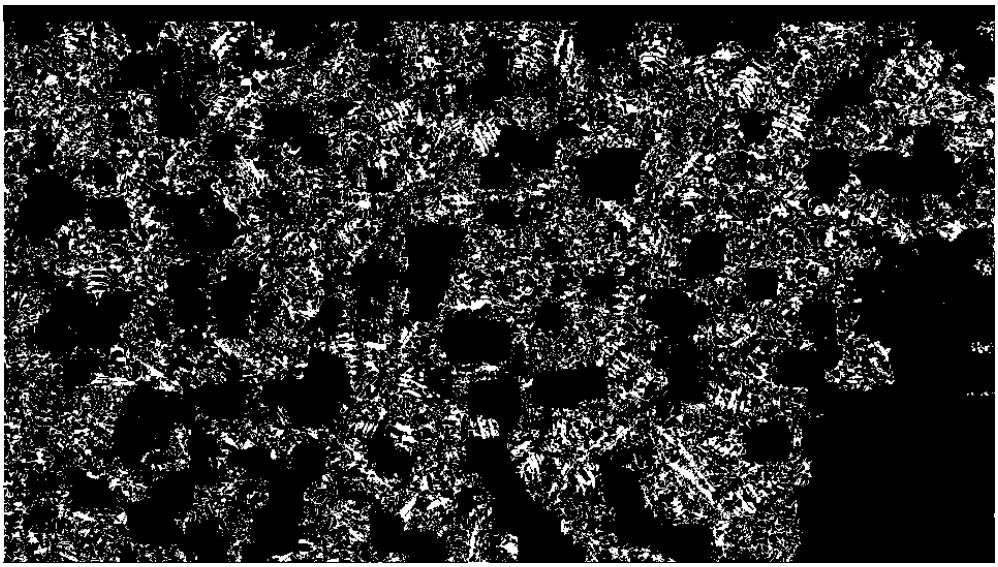
7.1 Klasické metody strojového vidění

Za použití procesu popsaného v sekci 6.1 byly nalezeny vhodné kombinace filtrovacích funkcí a segmentačních metod, jak je popsáno v tabulce 7. Je však nutné podotknout, že globální prahování se nejeví jako vhodná metoda segmentace z důvodu silně variabilního osvětlení.

Filtrace	Segmentace
Bilaterální filtr	Prahování
Filtrace medianem	Globální prahování
Filtrace medianem	Canny

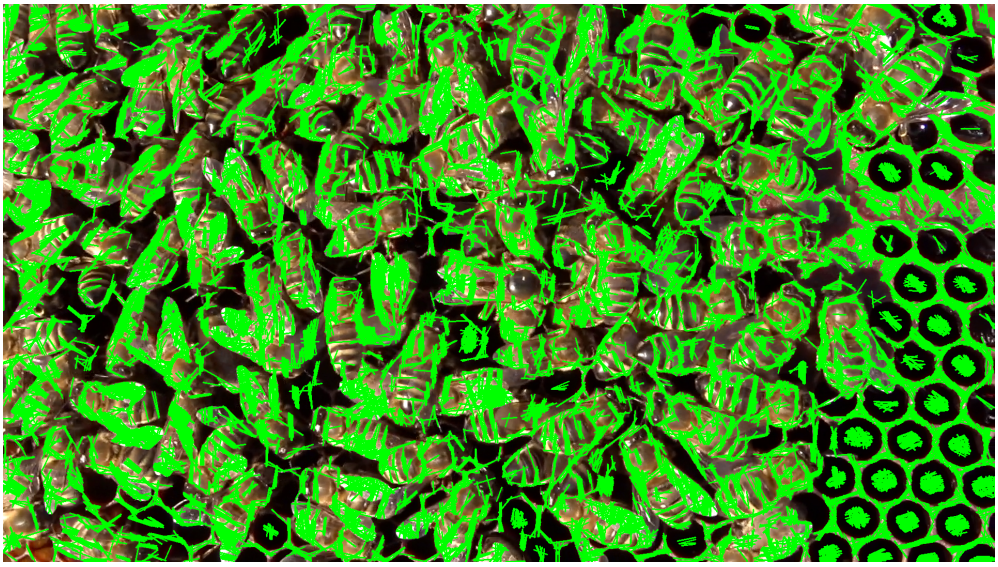
Tabulka 7: Kombinace funkcí filtrace a segmentace.

Od vlastní implementace segmentační metody s pomocí gradientů bylo upuštěno, jelikož na výstupu generovala téměř pouze šum, viz obrázek 10.



Obrázek 10: Výstup vlastní implementace za použití výpočtu gradientů.

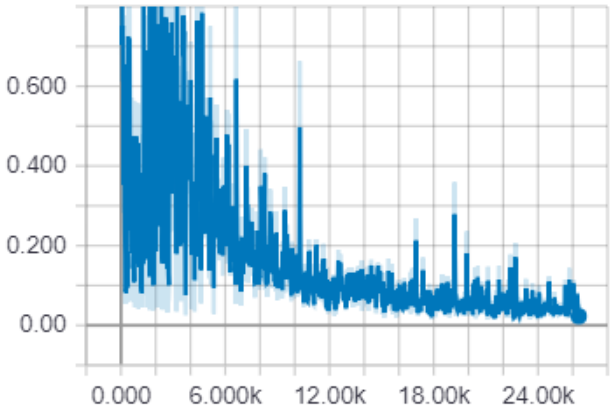
Výstupy pravděpodobnostní varianty Houghovy transformace jsou téměř ve všech zkoumaných parametrizacích též značně zašuměné, což je i důvod, proč v této oblasti nebylo zkoumání dále rozvíjeno. Jako příklad je uveden obrázek 11.



Obrázek 11: Výstup houghovy transformace.

7.2 Neuronové sítě

Model Inception byl podle návodu v [23] trénován dokud se chyba sítě nezačla držet pod hodnotou 0,1; což odpovídalo zhruba 25000 kroků. Průběh učení je graf na obrázku 12. Validována byla síť s použitím nahrávky, na které se vyskytuje značená matka a tedy nepoužité v trénovacím datasetu. 17,6 % doby nahrávky byla matka správně lokalizována. 8,2 % doby nahrávky síť dávala false positive výstup. Dlužno podotknout, že momenty kdy byla matka správně označena byly delší a souvislé, kdežto false positive indikace nebyly delší než několik snímků. Přesto se výsledek zdá značně neuspokojivý.

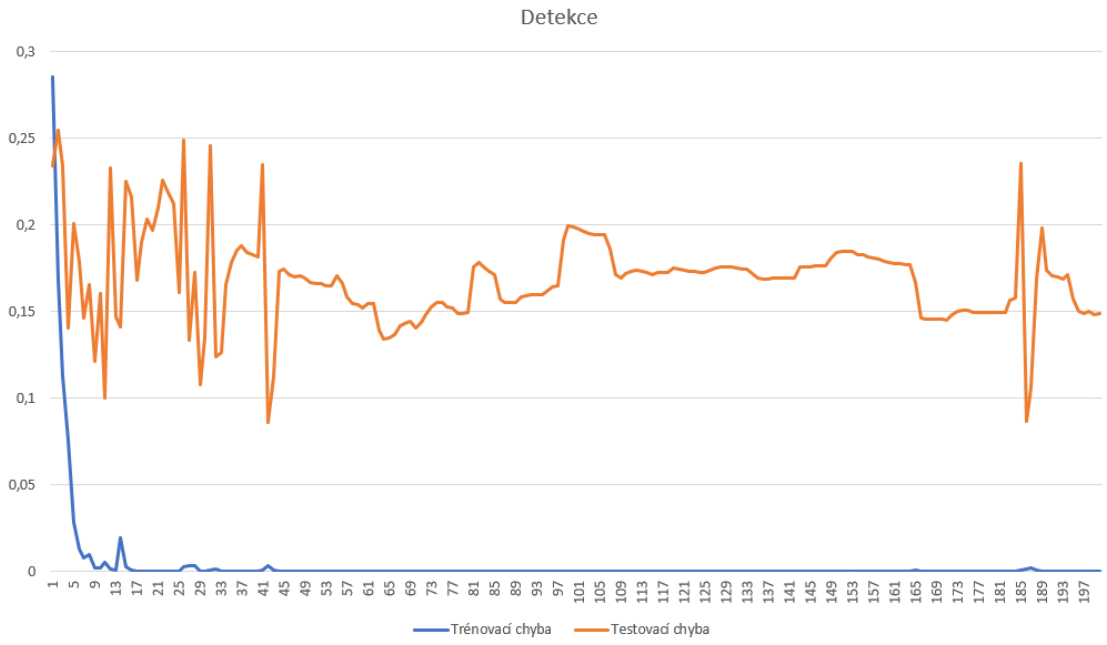


Obrázek 12: Průběh učení transfer learningu Inception. Graf je výstupem vizualizační utility Tensorboard.

Mezi sítěmi typu VGG dosáhla nejlepšího validačního výsledku síť číslo 4, jak je vidět v tabulce 8. Ačkoliv validační chyba se s narůstající hloubkou sítě zmenšuje, poslední nejhlubší síť zaznamenává propad. Zdůvodněním může být snížení rozlišovací schopnosti sítě způsobené zmenšním výstupu poslední konvoluční vrstvy.

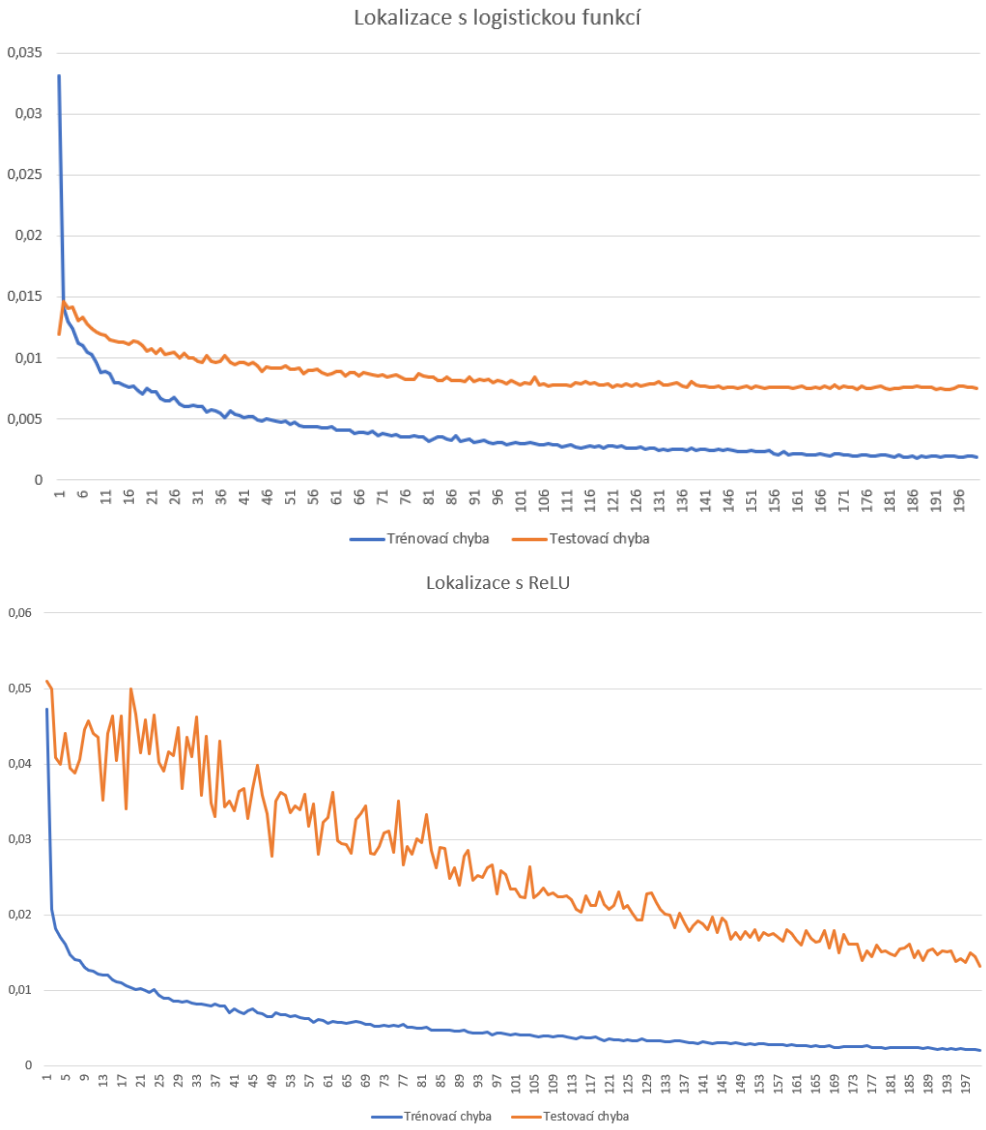
Síť	Trénovací chyba	Testovací chyba	Validační chyba
Detekce	0,0002	0,1346	0,0603
Lokalizace logistická	0,0020	0,0075	0,0025
Lokalizace ReLU	0,0021	0,0132	0,0064
Lokalizace 3	0,0006	0,0067	0,0015
Lokalizace 4	0,0007	0,0069	0,0014
Lokalizace 5	0,0010	0,0066	0,0023

Tabulka 8: Srovnání chyb jednotlivých sítí v bodě jejich nejlepšího uloženého řešení.

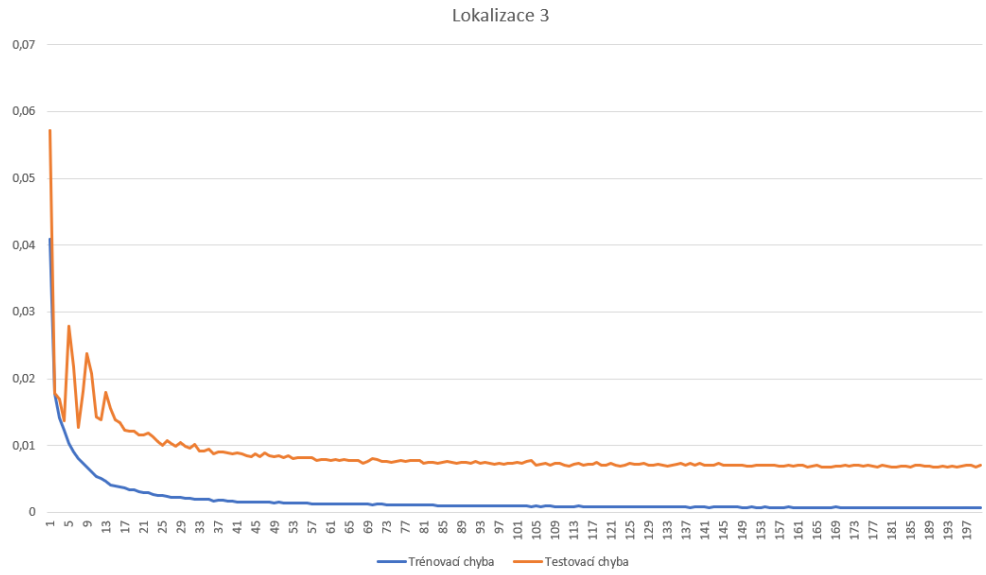


Obrázek 13: Průběh učení detekovací sítě.

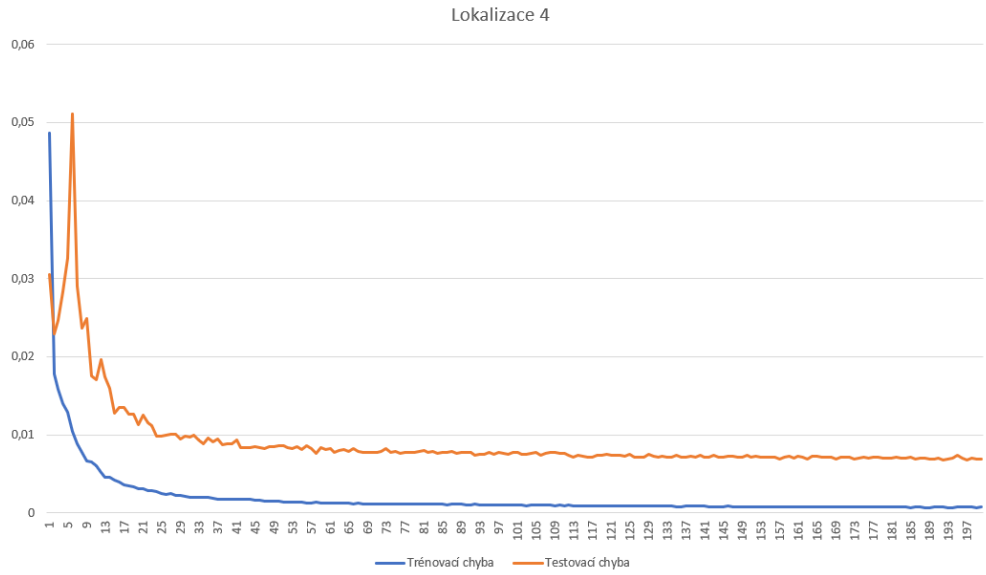
Na obrázcích 14 až 17 je možno pozorovat graf průběhu učení lokalizačních sítí. Ve všech případech dochází k určité míře přeučení.



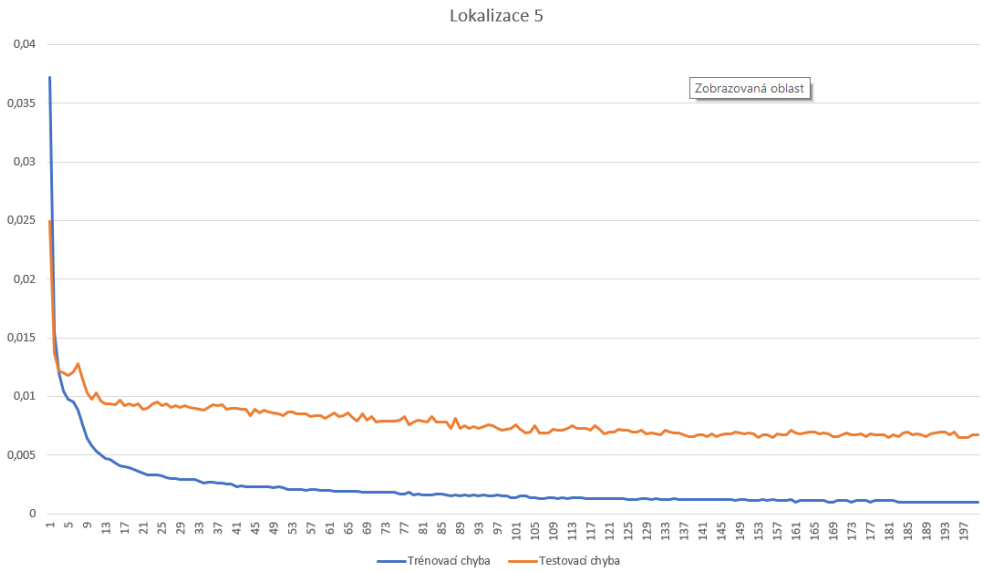
Obrázek 14: Srovnání průběhu učení prvních lokalizačních sítí.



Obrázek 15: Průběh lokalizace 3.

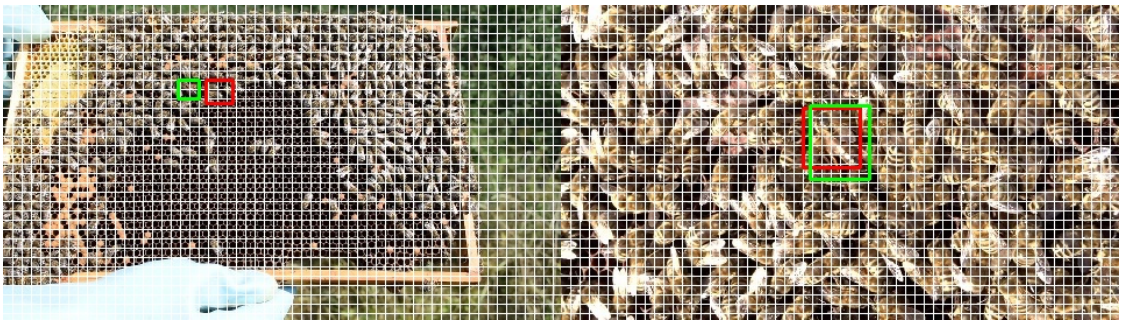


Obrázek 16: Průběh lokalizace 4.



Obrázek 17: Průběh lokalizace 5.

O přesnosti trénovaných VGG sítí se dá jen stěží hovořit, jelikož na většině validačních vzorů nemá detekovaná oblast průnik se správným výsledkem, pouze se těsně přibližuje. Vzory, kde průnik existuje, ty na kterých je včelí matka velká a dobře viditelná jsou však detekovány téměř bezchybně. Příklady těchto případů lze pozorovat na obrázku 18.



Obrázek 18: Ilustrace predikcí sítě 4. Mřížka zobrazuje schopnost rozlišení sítě, zelený rámeček značí polohu matky, červený značí predikci.

8 Závěr

Hlavním cílem práce bylo prozkoumat metody detekce včelí matky a v ideálním případě zpracovat implementaci, která umožní její lokalizaci.

V oblasti klasických metod strojového vidění se nepodařilo takovou implementaci najít, nebyla tedy potvrzena ani vyvrácena původní hypotéza o možnosti detekce včelí matky detekcí anomálií v pohybu ostatních včel. Byly prozkoumány některé funkce knihovny OCV a vytvořen proces užitečný k ladění jejich parametrizace. Odlesky na křídlech včel způsobují nepříjemný šum, který vytváří výkyvy v histogramu a způsobují tak, že po dokončení segmentace zůstávají vždy v obraze nechtěné oblasti navíc. Námětem na další práci v této oblasti tedy může být zkoumání metod, které odlesky potlačují, případně vymyslet zpracování výstupu, které by si s tímto šumem dovedlo poradit.

V oblasti neuronových sítí se podařilo dosáhnout dílčího úspěchu, kdy nejlepší dosažené řešení detekuje správně jen v některých případech. Při použití současé implementace pro mobilní aplikaci by byl uživatel nucen telefonem skenovat rámky vytažené ze včelího úlu po menších částech tak, aby bylo zajištěno, že případně snímaná matka má v obraze dostatečnou velikost. Zlepšení by pravděpodobně bylo možno dosahnout navržením architektury, která má větší hloubku za zachování nebo dokonce zlepšení její schopnosti rozlišení. Dále by bylo vhodné pro zmírnění přeučení nasbírat další materiál a rozšířit v současné době relativně malý dataset.

Reference

[1] BIENEFELD, Kaspar. Včelařství krok za krokem: pro milovníky krásného koníčka. 2. vyd. Přeložil Anna ŠTORKÁNOVÁ. Líbeznice: Víkend, 2010. ISBN 978-80-7433-023-0

[2] BERÁNEK, Vladimír. Včelařská encyklopedie. 2. přeprac. a rozš. vyd. Praha: Státní zemědělské nakladatelství, 1956. Živočišná výroba (Státní zemědělské nakladatelství)

[3] HLAVÁČ, Václav a Milan ŠONKA. Počítačové vidění. Praha: Grada, 1992. ISBN 80-85424-67-3

[4] GONZALEZ, Rafael C a Richard E WOODS. Digital image processing. 2nd ed. Upper Saddle River, N.J.: Prentice Hall, c2002. ISBN 9780201180756

[5] SHAPIRO, Linda G a George C STOCKMAN. Computer vision. Upper Saddle River, NJ: Prentice Hall, 2001. ISBN 978-0130307965.

[6] CANNY, John. A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence [online]. 1986, PAMI-8(6), 679-698 [cit. 2018-12-02]. DOI: 10.1109/TPAMI.1986.4767851. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4767851

[7] GREEN, Bill. Edge Detection Tutorial. Drexel University [online]. 2002 [cit. 2018-12-02]. Dostupné z: https://web.archive.org/web/20160324173252/http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/can_tut.html

[8] SOBEL, Irwin. (2014). An Isotropic 3x3 Image Gradient Operator. Presentation at Stanford A.I. Project 1968.

[9] HOUGH, P. V. C. Method and means for recognizing complex patterns. USA. US3069654A Patent. Zapsáno 18.12.1962.

[10] DUDA, Richard O. a Peter E. HART. Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM [online]. 15(1), 11-15 [cit. 2018-12-03]. DOI: 10.1145/361237.361242. ISSN 00010782. Dostupné z: http://portal.acm.org/citation.cfm?doid=361237.361242

[11] Neural Network. DeepAI [online]. Santa Barbara, CA: DeepAI, 2018 [cit. 2018-12-05]. Dostupné z: https://deepai.org/machine-learning-glossary-and-terms/neural-network

[12] Machine Learning. DeepAI [online]. Santa Barbara, CA: DeepAI, 2018 [cit. 2018-12-05]. Dostupné z: https://deepai.org/machine-learning-glossary-and-terms/machine-learning

[13] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep Learning [online]. MIT Press, 2016 [cit. 2018-12-05]. Dostupné z: http://www.deeplearningbook.org

[14] NIELSEN, Michael A. Neural Networks and Deep Learning [online]. Determination Press, 2015 [cit. 2018-12-06]. Dostupné z: http://neuralnetworksanddeeplearning.com/index.html

[15] FUMO, David. A Gentle Introduction To Neural Networks Series - Part 1. Towards Data Science [online]. 4.8.2017 [cit. 2018-12-06]. Dostupné z: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

[16] LECUN, Y., B. BOSER, J. S. DENKER, D. HENDERSON, R. E. HOWARD, W. HUBBARD a L. D. JACKEL. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation [online]. 1989, 1(4), 541-551 [cit. 2018-12-07]. DOI: 10.1162/neco.1989.1.4.541. ISSN 0899-7667. Dostupné z: http://www.mitpressjournals.org/doi/10.1162/neco.1989.1.4.541

[17] KRIZHEVSKY, Alex, Ilya SUTSKEVER a Geoffrey E. HINTON. ImageNet classification with deep convolutional neural networks. Communications of the ACM [online]. 2017, 60(6), 84-90 [cit. 2018-12-07]. DOI: 10.1145/3065386. ISSN 00010782. Dostupné z: http://dl.acm.org/citation.cfm?doid=3098997.3065386

[18] RUSSAKOVSKY, Olga, Jia DENG, Hao SU, et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision [online]. 2015, 115(3), 211-252 [cit. 2018-12-07]. DOI: 10.1007/s11263-015-0816-y. ISSN 0920-5691. Dostupné z: http://link.springer.com/10.1007/s11263-015-0816-y

[19] SIMONYAN, Karen a Andrew ZISSERMAN. Very Deep Convolutional Networks for Large-Scale Image Recognition [online]. 2014 [cit. 2018-12-07]. Dostupné z: https://arxiv.org/abs/1409.1556

[20] SZEGEDY, Christian, WEI LIU, YANGQING JIA, et al. Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 2015, 2015, s. 1-9 [cit. 2018-12-08]. DOI: 10.1109/CVPR.2015.7298594. ISBN 978-1-4673-6964-0. Dostupné z: http://ieeexplore.ieee.org/document/7298594/

[21] LIN Min, Qiang CHEN a Shuicheng YAN. Network In Network [online]. 2013 [cit. 2018-12-08]. Dostupné z: https://arxiv.org/abs/1312.4400

[22] YOSINSKI, Jason, Jeff CLUNE, Yoshua BENGIO a Hod LIPSON. How transferable are features in deep neural networks?. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 [online]. Montreal, Canada: MIT Press, 2014, s. 3320-3328 [cit. 2018-12-08]. NIPS'14. arXiv:1411.1792 [cs.LG]. Dostupné z: https://arxiv.org/abs/1411.1792

[23] EDJEELECTRONICS. How To Train an Object Detection Classifier for Multiple Objects Using TensorFlow (GPU) on Windows 10 [online]. Seattle, WA, 2018 [cit. 2018-12-08]. Dostupné z: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10