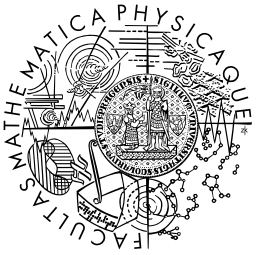


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Markéta Machalová

Detekce kleštíka včelího pomocí počítačového vidění

Katedra algebry

Vedoucí bakalářské práce: Ing. Adam Novozámský, Ph.D.

Studijní program: Matematika

Studijní obor: Matematika pro informační technologie

Praha 2021

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Na tomto místě bych chtěla poděkovat Ing. Adamu Novozámskému, Ph.D., za vedení práce, cenné rady, připomínky a nové vědomosti i z oblasti včelařství.

Děkuji kamarádce Ivě Hrabánkové za kvalitní korekturu pravopisu, gramatiky a stylistiky. A v neposlední řadě děkuji Bohu, rodině a přátelům za podporu.

Název práce: Detekce kleštíka včelího pomocí počítačového vidění

Autor: Markéta Machalová

Katedra: Katedra algebry

Vedoucí bakalářské práce: Ing. Adam Novozámský, Ph.D., Ústav teorie informace a automatizace AV ČR, v. v. i.

Abstrakt: Bakalářská práce se zabývá návrhem a popisem nástroje, který dokáže usnadnit proces monitorování varroázy. Využitím metod zpracování obrazu detekuje na fotografii spadové podložky z úlu jednotlivé kleštíky.

Pro lepší rozlišení jsme využili několika snímků z jedné spadové podložky, které jsme následně pomocí obrazové registrace sešili do jednoho obrázku. V první fázi jsme se zaměřili na využití klasických metod zpracování obrazu, jež dokážou detekovat kleštíka pouze na základě aproximace těla kleštíka parametricky popsatelnou křivkou. V druhé fázi jsme využili výkonnější konvoluční neuronovou síť. Při každé z 500 epoch trénování jsme si ukládali parametry úspěšnosti. Na závěr jsme do naučené sítě vložili testovací sadu obrázků a porovnávali je s očekávanými výstupy.

Práce obsahuje teoretický popis algoritmů a metod, jejich využití v našem detektoru a interpretaci výsledků.

Klíčová slova: automatizace, zpracování obrazu, sešívání snímků, detekce objektů

Title: Detection of Varroa Destructor Using Computer Vision

Author: Markéta Machalová

Department: Department of Algebra

Supervisor: Ing. Adam Novozámský, Ph.D., Institute of Information Theory and Automation of the CAS

Abstract: The bachelor thesis deals with the design and description of a tool that can simplify the process of monitoring varroosis. Using image processing methods, it detects individual mites from the picture of the bottom board in the beehive.

For better image resolution we used several images from one bottom board. Those images were then stitched into one image using image registration. In the first part we focused on the use of classical image processing methods that can detect varroa only on the basis of the approximation of the body of the mite with a parametrically descriptive curve. In the second part we used a more powerful convolutional neural network. During each of the 500 training epochs we saved the parameters of success. Finally, we inserted a test data into the trained network and compared it with expected outputs.

The thesis contains a theoretical description of algorithms and methods, their use in our detector, and interpretation of results.

Keywords: automation, image processing, image stitching, objects detection

Obsah

Úvod

Kleštík včelí (*Varroa destructor*) je zástupcem ektoparazitů napadajících včely východní a včely medonosné. Jak samotný latinský název napovídá, kleštíci představují opravdové *destruktory* včelstva [1]. Za necelé století se rozšířili z Asie na tři další kontinenty [2]. A jelikož napadají právě nenahraditelného opylovače a producenta medu, včelaři musí být na pozoru a pravidelně monitorovat stav svých úlů.

Cílem této bakalářské práce je navrhnout a popsat nástroj, který by na zachycených snímcích spadové podložky v úlu dokázal s co největší přesností rozpoznat samotné kleštíky. Tím by se včelaři usnadnil proces monitorování varroázy (onemocnění včel, které způsobuje kleštík včelí). Práce byla motivována projektem *Vývoj pokročilého komplexního monitorovacího systému*. Díky tomuto projektu jsme měli možnost získat data od většího množství včelařů, kteří spadové podložky ve svých úlech zachytili fotoaparátem na mobilním telefonu.

Text práce je členěn do pěti základních kapitol. V první části se zaměříme na detekovaný objekt, tedy na kleštíka včelího. Detailněji si vysvětlíme, proč je monitoring varroázy tak zásadní a jak je tento proces pracný bez využití počítačového vidění.

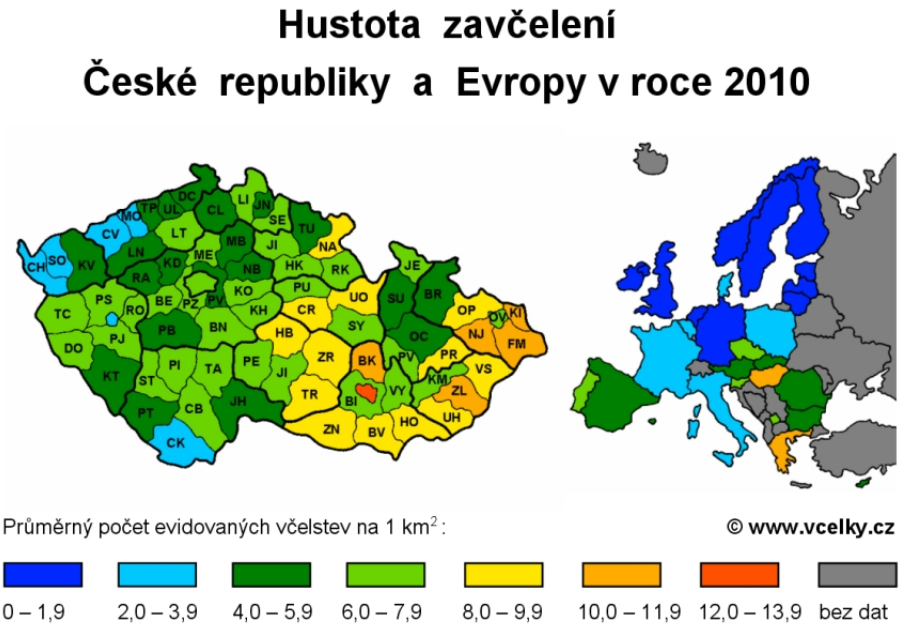
Zbývající kapitoly se už budou zabývat výhradně zpracováním obrazu. Nejprve si vysvětlíme, proč je užitečné obrázek spadové podložky vytvořit z více snímků. Ukážeme si, jak snímky sešít dohromady a jaké různé algoritmy k tomu můžeme využít, a to včetně schematického popisu implementace.

Následující dvě kapitoly obsahují dva základní přístupy k detekci objektů. První z nich využívá klasických metod zpracování obrazu, druhý pak konvoluční neuronovou síť. Na úplný závěr aplikujeme všechny poznatky z předešlých kapitol na detekci kleštíka včelího ze spadových podložek v úlu. Podíváme se na sešívání snímků podložky, proces anotace dat, implementaci konkrétní sítě a klasifikaci dat na naučené síti.

1. Včelstvo a jeho největší hrozba

1.1 Zavčelení České republiky

Česká republika patří mezi nejzavčelenější státy světa, tedy mezi státy s největším průměrem počtu včelstev na 1 km^2 svého území. V Evropě dokonce obsadila v roce 2010 třetí místo v žebříčku zemí s největší hustotou zavčelení [3]. Obrázek 1.1 ukazuje, že mezi jednotlivými regiony jsou značné rozdíly. Zatímco Mostecko má v posledních dvaceti letech v průměru 2–4 včelstva na 1 km^2, na Brněnsku dosahují hodnoty až pětinásobku. Konkrétně v okrese Brno-město bylo v roce 2015 průměrně 16,2 včelstev na 1 km^2.



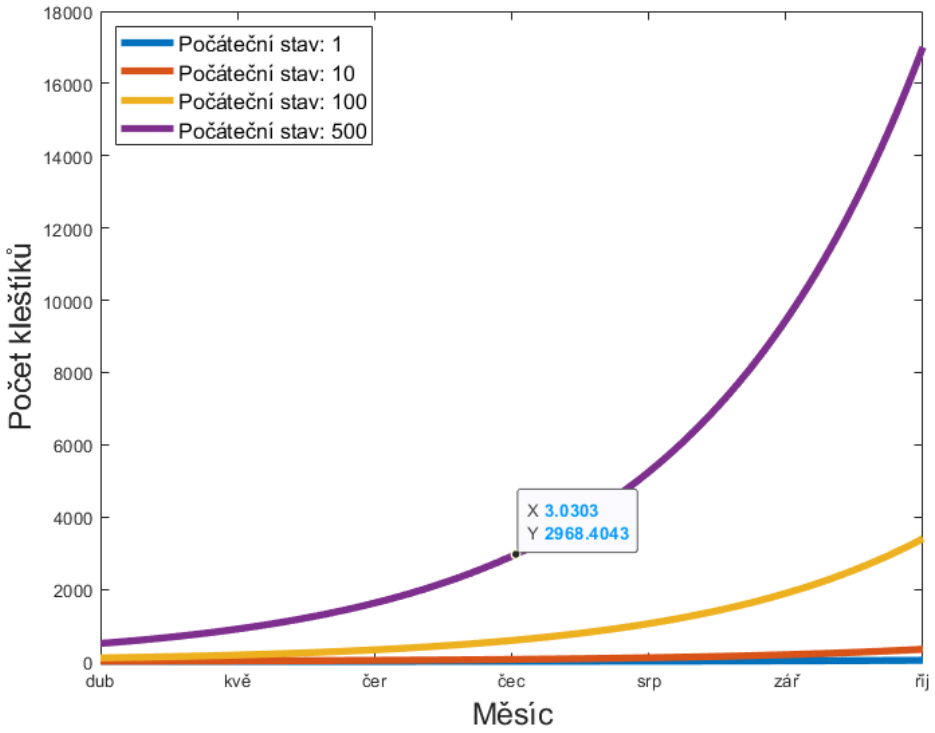
Obrázek 1.1: Hustota zavčelení České republiky v kontextu Evropy [3]

Dle poslední zprávy o zemědělství ČR dochází k neustálému nárůstu počtu včelařů a včelstev. V roce 2019 vystoupal počet včelařů na 62,9 tisíc a počet včelstev na 685 tisíc [4]. A není překvapivé, že s nárůstem počtu včelstva přibývá i jejich parazitů. V současné době je největší hrozbou právě kleštík včelí, který způsobuje onemocnění s názvem varroáza. V následujících sekcích se proto zaměříme na popis tohoto roztoče a ukážeme si, jak co nejúčinněji bojovat s varroázou.

1.2 Populační dynamika kleštíka včelího

Kleštík včelí patří mezi parazity, kteří se velmi rychle rozmnožují. Během jediného dvoutýdenního reprodukčního cyklu vznikne v průměru 1,45–2,2 nových samiček [5]. Nižší hodnota odpovídá reprodukci v dělničí buňce a vyšší naopak v trubčí buňce.

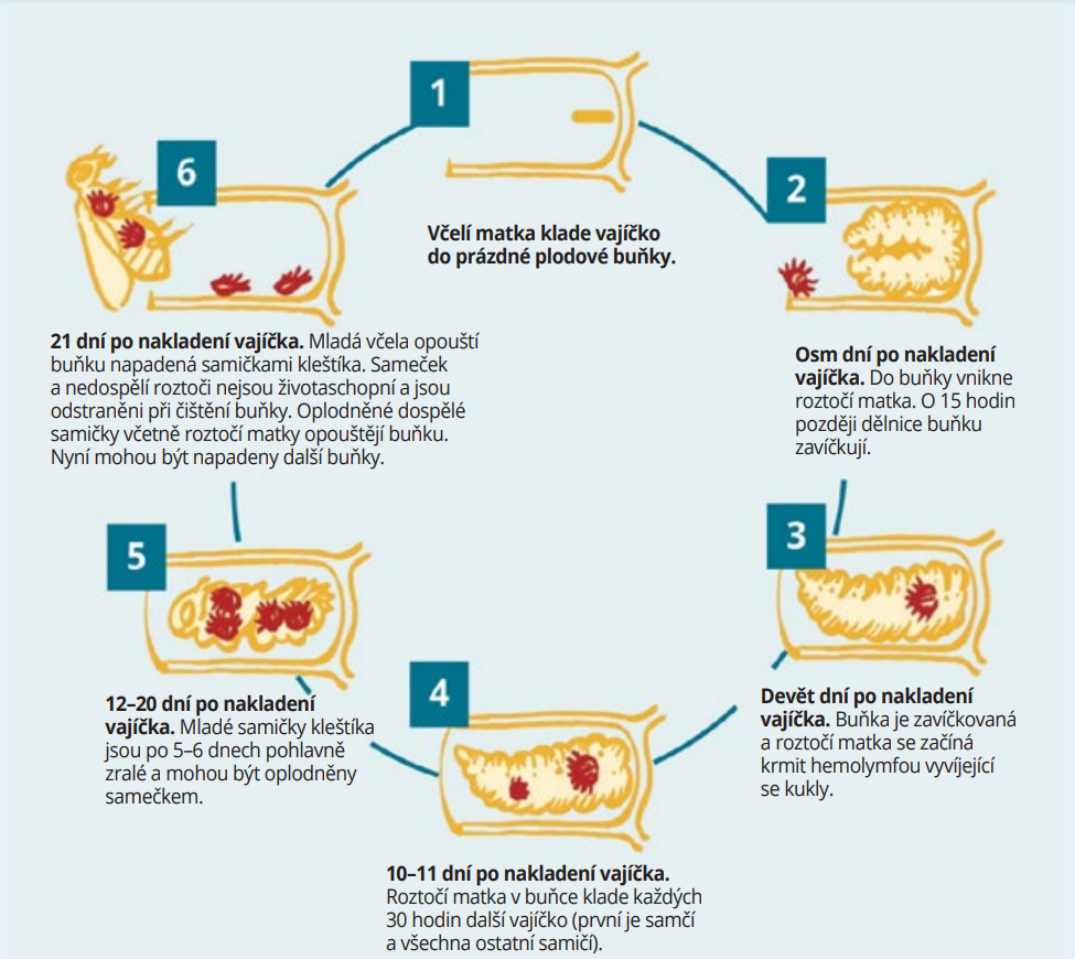
Jak ukazuje graf v obrázku 1.2, populace roztoče se téměř zdvojnásobí za jediný měsíc, přičemž v případě, kdy dochází k přenosu kleštíka mezi sousedními včelstvy, bývá vývoj ještě rychlejší. Zároveň z grafu vyplývá, že počet jedinců na začátku jarního období značně ovlivňuje konečný nárůst populace. Platí, že čím více parazitů, tím rychlejší růst populace. Už v červenci je kleštíků 6× víc oproti dubnu [6]. I proto v případě nálezu 3 a více kleštíků na jedné spadové podložce mají čeští včelaři povinnost provést do 15. dubna předjarní léčebné ošetření včelstev [7].



Obrázek 1.2: Model vývoje populace kleštíka.[1]

Většina kleštíků ve včelstvu se nachází v zavíčkovaných buňkách. Roztočí matka se zde živí hemolymfou z kukly a každých 30 hodin klade jedno vajíčko (přičemž první je samčí a zbylá samičí). Po 5–6 dnech jsou samičky pohlavně zralé a mohou být oplodněny samečkem. Mimo plodovou buňku dokážou přežít pouze oplodněné samičky, tedy na závěr celého cyklu opouští buňku napadená včela spolu se samičkami kleštíka, naopak samečci a neoplodněné samičky zahynou ještě v buňce. Podrobnější popis celého reprodukčního cyklu je k nahlédnutí v obrázku 1.3.

[1] Graf vytvořen na základě dat z [6, 8]. Odpovídá funkcím $y = a \cdot 1{,}8^x$, kde a představuje počáteční stav. Základ 1,8 naopak vychází z průměrného přírůstku.



Obrázek 1.3: Reprodukční cyklus [5]

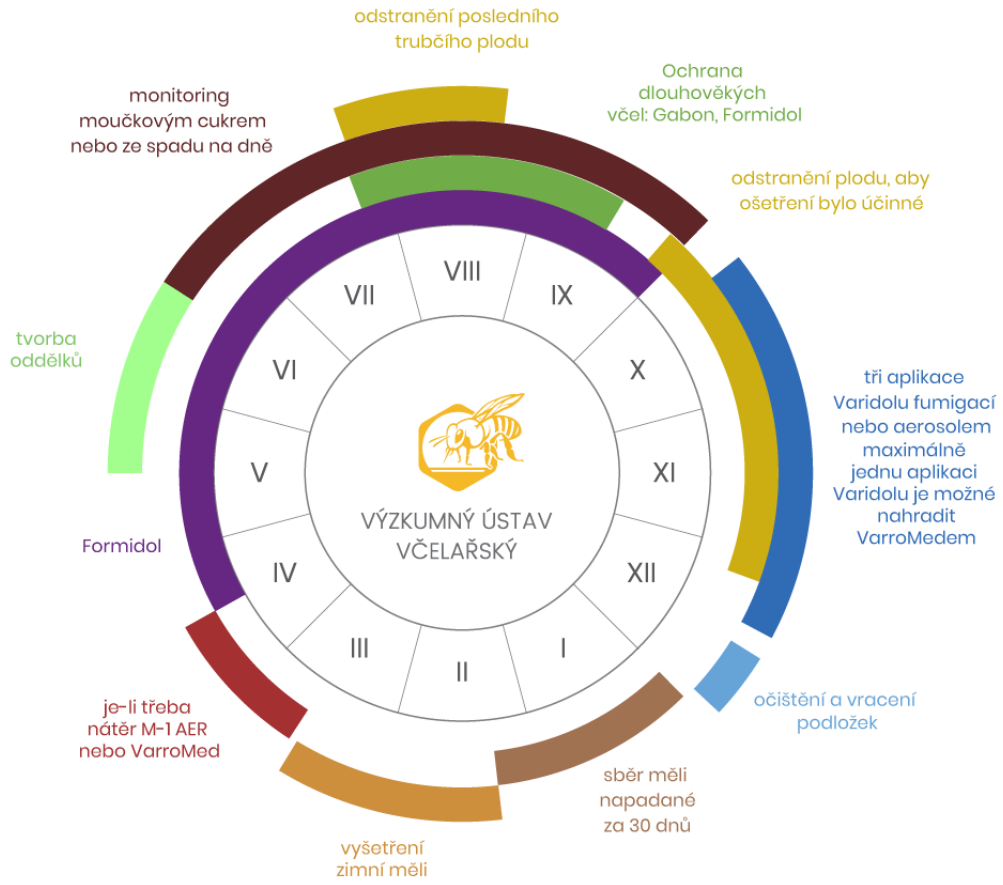
1.3 Varroáza a její léčba

Závažnost varroázy spočívá ve snižování odolnosti včel vůči škůdcům, v negativním ovlivnění opylování rostlin, v riziku vzniku dalších chorob a v neposlední řadě ve snižování výkonnosti včel, které vede ke snížení produkce medu. Při nedostatečné léčbě může dojít až k úhynu včelstva [6].

Kvůli závažnosti tohoto onemocnění vznikl závazný Metodický pokyn Státní veterinární správy pro chovatele včel k prevenci a tlumení varroázy. Ten obsahuje nejen celostátně povinná ošetření, ale i různá další doporučení [9]. Základní kroky léčby, seřazené dle kalendářního roku, jsou uvedeny v obrázku 1.4 a popsány v následujících odstavcích.

- **Vyšetření zimní měli**[2]: Vzorky z podložky podrobíme vyšetření na varroázu (tedy manuálnímu počítání jednotlivých kleštíků) a na základě výsledků zjistíme, zda je potřeba zahájit léčbu či nikoliv. Při odběru je potřeba dbát na to, abychom z měli odstranili všechny včely a následně odebrali veškerou zimní měl. V opačném případě mohou být výsledky vyšetření zkreslené.

[2]Měl je odpad na dně úlu. Obsahuje biologické zbytky z včelstva a další odpad vyprodukovaný během budování a ošetřování plástů.



Obrázek 1.4: Roční cyklus monitorování a léčby varroázy [9]

- **Nátěr víčka plodových buněk**: Po vyšetření zimní měli se v případě indikace provádí nátěr zavíčkovaných buněk plodu. Účinná látka proniká do buněk i mezi včely a dokáže tak napravit nedostatečné zimní ošetření včelstev.
- **Aplikace Formidolu**: V případě indikace aplikujeme v dubnu až červenci léčivý přípravek na bázi organických kyselin, například Formidol (kyselina mravenčí). Kromě ničení roztočů na včelách dokáže stimulovat u včel jejich čístící pud [10].

  Formidol patří mezi žíraviny, aplikaci lze tedy provést pouze ve chvíli, kdy se v úlu nenachází med určený k lidské konzumaci. Zároveň je nutné dbát i na vlastní bezpečnost a vybavit se ochrannými pomůckami.
- **Monitoring**: Viz sekce 1.4.
- **Odstranění posledního trubčího plodu**: Odstraněním posledního trubčího plodu docílíme značné redukce počtu kleštíků, což plyne z toho, že kleštíci dávají přirozeně přednost trubčímu plodu. Konkrétně výskyt varroázy u trubčího plodu je 8–10× pravděpodobnější než u buněk dělnic [11].

  Na obdobném principu je založen i monitoring ze zavíčkovaného trubčího plodu. Využitím odvíčkovací vidličky lze objevit případné vysoké napadení.

Výhodou je, že s tímto typem monitoringu lze začít už během jara, zatímco u ostatních metod se začíná až v červenci.

- **Ochrana dlouhověkých včel**: Kromě zmíněného Formidolu se pro ochranu včelstva používají i proužky Gabon. Pokud se včela dostane do kontaktu s povrchem proužku, dojde k vyhubení kleštíků přisátých na jejím těle. Navíc se účinná látka z Gabonu šíří dál a dokáže tak zneškodnit kleštíky na včelách, které se teprve líhnou [12].

  Je však potřeba začít s vkládáním proužků do úlu včas, nejpozději začátkem srpna. Rovněž je nutné kontrolovat, zda ochrana funguje, neboť riziko rezistence vůči amitrazu či pyrethroidu (léčivým látkám) je čím dál větší. Proto se doporučuje neaplikovat Gabon každý rok, ale některé roky, kdy je výskyt kleštíka slabší, vynechat.

- **Odstranění posledního plodu, klíckování matek**: Odstranění posledního plodu probíhá stejně jako u výše zmíněného trubčího plodu. Rozdíl je pouze v tom, že na konci léta se v úlu vyskytuje jen plod dělnic.

  Klíckování neboli izolace včelích matek slouží k minimalizaci počtu roztočů v období letních prázdnin. Tento proces je důležitý z toho důvodu, že podzimní léčení mnohdy není dostačující [13]. Pro izolaci matek slouží *izolátor*, do kterého zkraje léta vložíme matku, rámek s otevřeným plodem a rámek s mezistěnou. Po přibližně jedenácti dnech je vložený plod zavíčkovaný, a tak jej odebereme a nahradíme prázdnou souší. Za dalších jedenáct dnů odebereme rámky a vypustíme matku. S odstupem jednoho dne začneme s ošetřením a ideálně i s monitoringem kleštíků.

- **Léčení**: Nejdůležitější období pro léčbu varroázy nastává v zimě. V tomto období jsou včelstva bez plodu a kleštíci parazitují jen na dospělých včelách. V léčbě se kombinují ošetření fumigací (distribuce léčivé látky na doutnající knot) a ošetření aerosolem.

- **Čištění podložek**: V prosinci se provádí očištění a vrácení podložek, přibližně dva týdny po posledním ošetření. Tím je ukončen roční cyklus léčby varroázy a pokračuje se opět s vyšetřením zimní měli.

1.4 Klasický monitoring varroázy

Jak již bylo zmíněno v úvodu, každý včelař musí pravidelně monitorovat varroázu. V této sekci se proto zaměříme na popis dvou klasických okometrických metod [14].

Nejpřesnější metodou je monitoring využívající posypu včel cukrem. Z plodového plástu sklepneme do nádoby včely. Přidáme moučkový cukr a pečlivě promícháme. Následně na menší síto vysypeme obsah nádoby (přes děrované víko, aby včely neuletěly). Tím se nám cukr přeseje, kleštíci zůstanou v sítu a bude je možné spočítat. Výzkumný včelařský ústav upozorňuje, že při počtu vyšším než 5, 10, nebo 15 (v závislosti na měsíci měření, pro 600 včel) je třeba co nejdřív naplánovat léčbu varroázy. V případě výskytu více než 25 kleštíků je třeba začít s léčbou neprodleně.

Druhým přístupem je monitoring ze spadu na dno. Ten spočívá v pravidelné kontrole spadové podložky, na kterou padají přirozeně uhynulí jedinci. Pro přesnější výsledky je potřeba monitoring provádět ideálně denně a také zabránit vstupu mravenců do úlu, aby spadlé kleštíky neodnášeli. Kritickou hodnotou počtu kleštíků na podložce jednoho úlu je pět a více spadlých jedinců během jednoho dne.

Obě tyto metody vyžadují dostatek času a trpělivosti při závěrečném počítání jedinců. Obzvlášť při vysoké koncentraci kleštíků je velmi snadné se přepočítat a tím získat zkreslené údaje o aktuálním stavu včelstva. Proto se zde nabízí využít právě moderních technologií, které nám mohou výrazně ulehčit práci. Následující kapitoly budou věnovány jednotlivým krokům při inovativním monitoringu pomocí počítačového vidění.

2. Sešívání snímků

Prvním krokem monitoringu pomocí počítačového vidění je sešívání snímků spadové podložky. Ačkoliv obrázek lze vytvořit pouze z jediného snímku, jeho rozlišení by vzhledem k velikosti kleštíka a rozlišení fotoaparátů na mobilních telefonech nebylo dostatečné.

Proces sešívání snímků se skládá ze čtyř základních kroků:

- Načtení obrázků
- Detekce příznaků
- Hledání korespondencí
- Odhad transformace mezi jednotlivými obrázky

Nejprve si představíme Harrisův detektor rohů, jehož základní myšlenky úzce souvisejí s dvěma nejužívanějšími metodami detekce příznaků a hledání korespondencí, které budou rovněž popsány. Na závěr se podíváme na výpočet transformace mezi obrázky.

2.1 Harrisův detektor rohů

Harrisův detektor rohů provádí detekci pomocí funkce D_h [15, 16]:

$$D_h(i,j) = \sum_x \sum_y w(x,y)[I(x+i,y+j) - I(x,y)]^2,$$

kde $I(x,y)$ představuje jeden pixel v obrázku I a $w(x,y)$ odpovídá filtru. Jako filtr bývá běžně volena Gaussova funkce $g(x,y) = \exp(\frac{x^2+y^2}{2\sigma^2})$
(σ určuje míru vyhlazení obrazu).

Nechť

$$H = \sum_x \sum_y w(x,y) \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix},$$

přičemž I_x a I_y jsou gradienty ve směru x a y. Potom využitím Taylorova rozvoje členu $I(x+i,y+j)$ aproximujeme funkci D_h následovně:

$$D_h = \begin{pmatrix} i & j \end{pmatrix} H \begin{pmatrix} i \\ j \end{pmatrix}.$$

Hledanou odezvu detektoru R získáme právě z determinantu a stopy matice H (*trace(H)*), konkrétně

$$R = \det(H) - k[\mathrm{trace}(H)]^2,$$

kde koeficient k leží v intervalu $\langle 0{,}04; 0{,}06 \rangle$. Rohové body pak odpovídají lokálním maximům funkce R.

Výhodou Harrisova detektoru je rotační a translační invariantnost. Nevýhodou je citlivost na změnu měřítka obrazu.

2.2 SIFT

Další metodou je *Scale-Invariant Feature Transform (SIFT)*. Tato metoda navazuje na Harrisův detektor. Jak již bylo zmíněno, tento detektor je velmi citlivý na změnu měřítka obrazu, proto není příliš vhodný pro sešívání obrázků různých rozměrů. Autoři metody SIFT se tedy pokusili tento problém vyřešit. [17, 18].

V prvním kroku rozostříme obrázek využitím Gaussova filtru, který má následující vzorec:

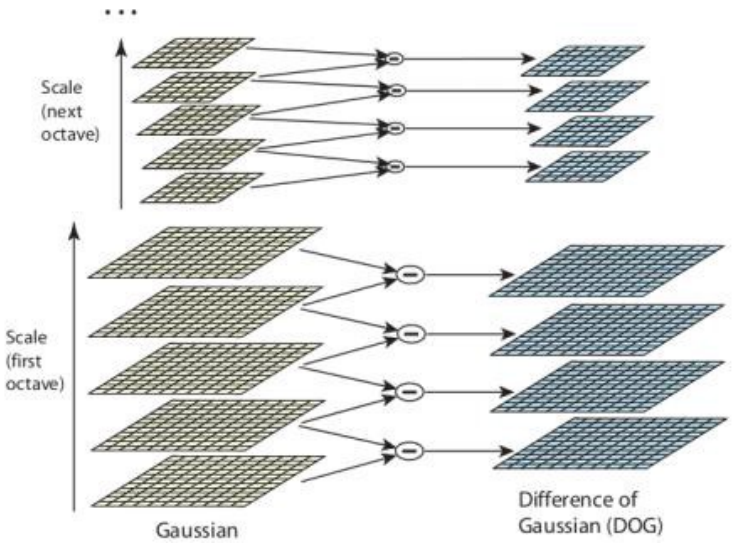
$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} \exp\Big(- \frac{x^2+y^2}{2\sigma^2}\Big).$$

Nechť $I(x,y)$ představuje vstupní obrázek a $*$ konvoluci. Potom rozostření obrázku odpovídá

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y).$$

Z jednotlivých rozostřených obrázků vytvoříme DoG (*Difference of Gaussians*), jež odpovídá rozdílu Gaussova rozostření stejného obrázku s různým σ:

$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma).$$

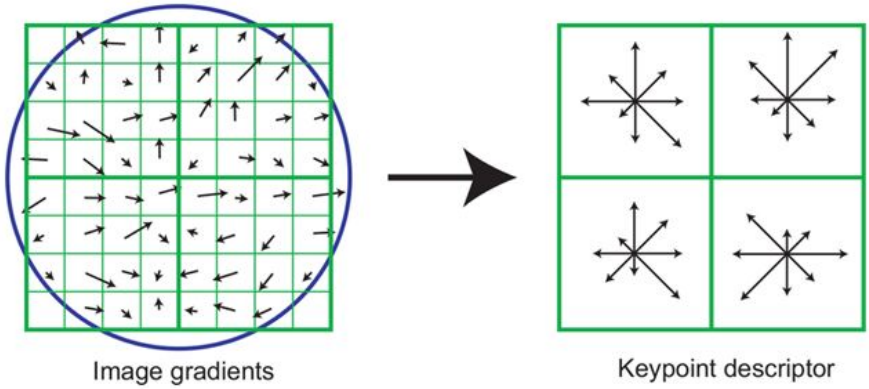


Obrázek 2.1: Znázornění scale-space [19]

Pro nalezení klíčových bodů aproximujeme LoG (*Laplacian of Gaussian*) porovnáváním se sousedními pixely. Konkrétně použijeme mřížku 3×3 a hodnotu funkce $D(x,y,\sigma)$ v prostředním bodě mřížky porovnáme s hodnotami funkce ve všech sousedních bodech. Těchto sousedů je 26, neboť porovnávání probíhá i se sousedními vrstvami.

V dalším kroku jsou vyřazeny body, které mají nedostatečný kontrast nebo které leží podél hran. U zbývajících bodů proběhne přiřazení orientace. Pro každý pixel se vypočítá velikost gradientu m a orientace α:

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \\ \alpha(x,y) &= \operatorname{arctg}\Big(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\Big). \end{aligned}$$



Obrázek 2.2: Tvorba deskriptorů [19]

Z gradientů se v okolí klíčového bodu vytvoří histogram. V něm se následně vybere jako orientace klíčového bodu lokální maximum. V případě detekování další vysoké hodnoty (větší než 80 % lokálního maxima) se vytváří klíčový bod na stejné pozici se stejným měřítkem daného lokálního maxima, jen s odlišnou orientací.

Další fáze se zaměřuje na konstrukci deskriptorů (obrázek 2.2). Deskriptory jsou 128-dimenzionální vektory složené z 16 histogramů pro 8 směrů. Tyto histogramy jsou tvořeny z oblastí velikosti 4×4.

Posledním krokem je hledání korespondencí mezi nalezenými body v obrázcích. K tomu slouží vypočtené deskriptory. Aplikuje se euklidovská vzdálenost a pro pixel v prvním obrázku se vybere z druhého obrázku pixel, jehož vzdálenost je nejnižší.

2.3 SURF

Vylepšením předchozí metody je algoritmus *Speeded-Up Robust Feature (SURF)*, který přináší viditelné urychlení procesu obrazové registrace a zároveň větší robustnost vůči transformacím, především rotační [20, 21].

Urychlení mimo jiné spočívá ve využití *integrálního obrazu*, tedy reprezentaci obrazu pomocí součtů předchozích pixelů. Konkrétně pro vstupní obrázek I s pixelem na pozici (x,y) získáme

$$I_{\sum}(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,j).$$

K detekci klíčových bodů využívá aproximaci Hessovy matice $H(x,y,\sigma)$ danou rovností

$$H(x,y,\sigma) = \begin{pmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{pmatrix},$$

kde (x,y) označuje souřadnice bodu obrazu o měřítku σ, $L_{xx}(x,\sigma), L_{xy}(x,\sigma)$ a $L_{yy}(x,\sigma)$ pak představují konvoluci druhé derivace Gaussiánu $\frac{\partial^2}{\partial x^2} g(\sigma)$ s původním obrazem v bodě (x,y).

Gaussián se skutečně ukazuje jako optimální pro analýzu měřítka a prostoru, přestože v praxi je potřeba funkci diskretizovat a oříznout. Výhoda rychlé konvoluce totiž převažuje nad nevýhodami, jako je například ztráta možnosti opakování pro některé rotace.

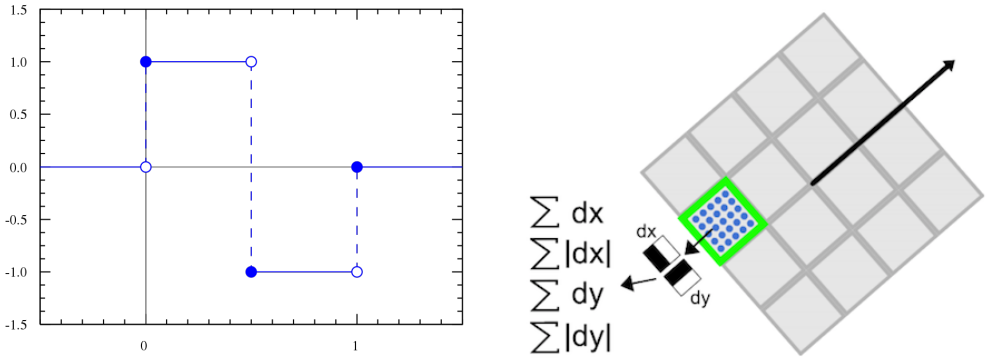
Výše zmíněné aproximace druhé derivace Gaussiánu získáme užitím filtru rozměru 9×9 s $\sigma = 1{,}2$. Označíme-li tyto aproximace D_{xx}, D_{xy} a D_{yy}, můžeme aproximovat determinant Hessovy matice následovně:

$$\det(H) = D_{xx}D_{yy} - (0{,}9D_{xy})^2.$$

Následně jsou obrázky vyhlazovány. Díky využití box filtru a integrálního obrazu není potřeba iterativně zmenšovat obrázek.

Nyní přichází na řadu tvorba deskriptorů. V prvním kroku se zaměříme na přiřazení orientace klíčových bodů. Nechť s označuje vrstvu (*scale*) vybraného bodu. Potom kolem klíčových bodů vytvoříme kruhy s poloměrem $6s$ a v nich vypočítáme odezvy *Haarovy vlnky* (viz obrázek 2.3) ve směru x a y.

Vytyčený kruh si pokaždé rozdělíme na šestiny a vypočteme v každé části součet vertikální a horizontální vlnkové odezvy. Výsledná orientace deskriptoru pak odpovídá maximálnímu součtu.



(a) Haarova vlnka [22] (b) Tvorba deskriptorů [19]

Obrázek 2.3: SURF

Na závěr sekce si popíšeme extrakci deskriptorů. Nejprve si kolem klíčových bodů vytvoříme okno o velikosti $20s$ a rozdělíme si ho na čtverce 4×4 (oblasti), ve kterých si ještě vyznačíme 25 rovnoměrně rozmístěných bodů. Pro každý takový bod využijeme zmíněnou Haarovou vlnku jakožto filtr velikosti $2s$ v horizontálním i vertikálním směru (viz obrázek 2.3). Poté ještě získané vlnkové odezvy sečteme s odezvami v rámci jedné oblasti – nejprve v absolutní hodnotě, poté bez absolutní hodnoty. Tím získáme výsledný deskriptor, který má tedy 4 prvky pro každou ze 16 oblastí.

Samozřejmě je ještě potřeba najít korespondenci mezi sešívanými obrázky. Zde se však přístup příliš neliší od metody SIFT.

2.4 RANSAC

Random Sample Consensus (RANSAC) představuje hojně využívaný algoritmus pro odhad matematického modelu z naměřených dat, která zahrnují i odlehlá pozorování (*outliery*). Jedná se o iterativní metodu, jež s rostoucím počtem iterací zaručuje větší pravděpodobnost spolehlivého modelu.

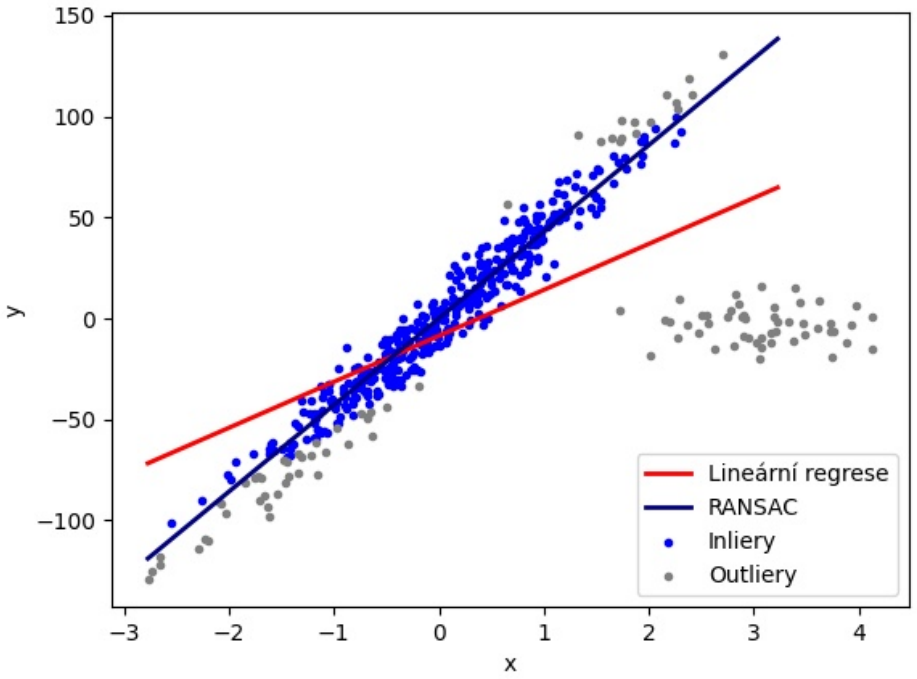
Označíme-li pravděpodobnost správného spojení příznaků obrazu p_i a počet korespondujících bodů ve vybrané sadě r, potom se pravděpodobnost nalezení správné transformace při n pokusech (obvykle 500) vypočte následovně [23]:

$$p_h = 1 - (1 - (p_i)^r)^n.$$

Z tohoto vzorce vyplývá, že při zvolení $n = 500$ a při $p_i = 0{,}5$ (jinak řečeno, pravděpodobnost výskytu *outlierů* je stejná jako pravděpodobnost výskytu *inlierů*) je pravděpodobnost

$$p_h \approx 1 - 10^{-14}.$$

Pro lepší představu o fungování algoritmu viz obrázek 2.4 demonstrující rozdíl mezi použitím lineární regrese (červená čára) a použití RANSACu (modrá čára).



Obrázek 2.4: RANSAC – náhodná data a proložení přímek

3. Detekce klasickými metodami

Nyní se dostáváme k samotné detekci. První způsob detekce je založený na klasických metodách zpracování obrazu, tedy bez využití neuronových sítí. Zároveň pracuje s předpokladem, že tělo kleštíka lze aproximovat elipsou.

V této kapitole si představíme *blob detektor* a jako klasifikátor *Houghovu transformaci pro detekci elips*. Ukážeme, jak algoritmy fungují a zda by mohly najít uplatnění při detekci kleštíka včelího.

3.1 Blob detektor

Prvním potenciálním nástrojem detekce je blob detektor. Detektor je založen na hledání objektů v obrázku podle zadaných parametrů. Postup detekce, jež byla implementována s využitím knihovny OpenCV, vypadá následovně [24]:

- Detektor převede vložený obrázek na několik binárních obrázků s rozdílnou prahovou hodnotou. Začíná na minimální prahové hodnotě a postupně hodnotu zvyšuje až na maximální. Všechny tyto hodnoty je možné programu zadat (včetně hodnoty, kterou postupně přičítáme).
- V každém obrázku vytvořeném v minulém kroku seskupí propojené bílé pixely.
- Následně nalezne středy těchto seskupených pixelů a ty, které jsou od sebe v menší vzdálenosti než zadaná minimální vzdálenost objektů, spojí do jednoho objektu.
- Detektor vrací středy a poloměry nalezených objektů.

Zmíněné objekty mohou mít nejrůznější vlastnosti. Někdy na nich příliš nezáleží, jindy potřebujeme detekovat něco konkrétního (například elipsy). Proto byl detektor vymyšlen tak, aby bylo možné mu nastavit parametry, podle kterých je schopen filtrovat objekty a vrátit pouze ty, které požadujeme.

Prvním parametrem je **obsah** objektu, u kterého můžeme nastavit minimum a maximum. Poté následuje **kruhovitost** K, která udává, jak je detekovaný objekt tvarově blízký kruhu. Definujeme ji vzorcem

$$K = \frac{4\pi S}{l^2},$$

kde S odpovídá obsahu a l obvodu. Při dosazení $S = \pi r^2$ a $l = 2\pi r$ zjistíme, že pro kruh $K = 1$. Tedy pokud hledáme elipsy, potřebujeme parametr nastavit přibližně mezi 0,8 a 1.

Třetím parametrem je **konvexita**, která je definována jako podíl obsahu objektu a obsahu jeho konvexního obalu. Jinými slovy, jedná se o poměr mezi detekovaným objektem a útvarem, který objekt zcela uzavírá a který je zároveň nejmenší s takovouto vlastností. V neposlední řadě můžeme nastavit i **poloměr setrvačnosti**, jenž zde odpovídá tomu, jak je objekt „protáhlý“.

Tento nástroj však není příliš vhodný pro detekci. Parametry detektoru sice můžeme různě nastavovat, ale v případě vyšší citlivosti detekuje větší množství

false positive, naopak při menší citlivosti neodhalí všechny kleštíky. Proto je potřeba využít klasifikátoru, který false positive v co největší míře odstraní. Nejprve vyzkoušíme Houghovu transformaci, metodu klasického zpracování obrazu.

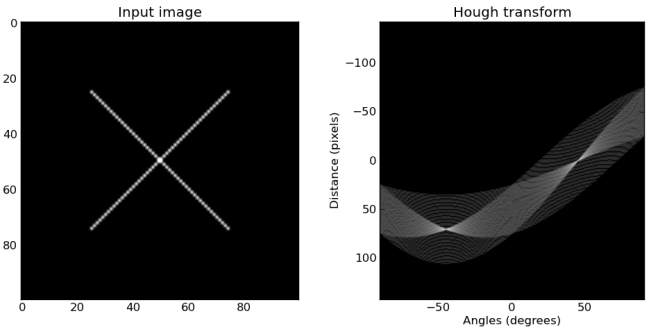
3.2 Houghova transformace

Popis metody

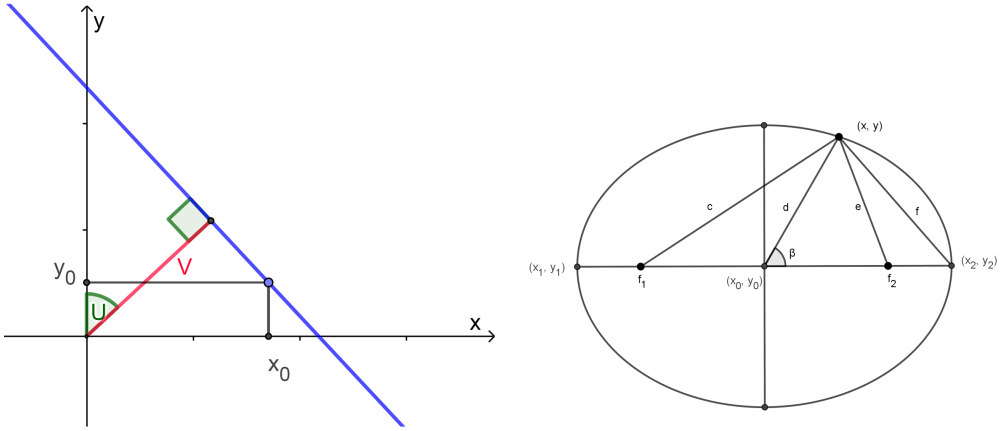
Houghova transformace je metoda pro detekci především parametricky popsatelných objektů (například přímek a kuželoseček) v obrázku. Základem transformace je *Houghův prostor*, do kterého transformujeme pixely z původního obrázku a na základě četnosti určujeme pozici detekovaného objektu.

Nejprve se zaměříme na transformaci pro detekci přímek, jež byla následně upravena pro návrh detekce složitějších objektů [25]. Základní vlastnosti Houghovy transformace jsou [26]:

- Bod v obrázku odpovídá sinusoidě v Houghově prostoru.
- Bod v Houghově prostoru odpovídá přímce v obrázku.
- Body ležící na stejné přímce v obrázku odpovídají křivkám, které procházejí společným bodem v Houghově prostoru.
- Body ležící na stejné křivce v Houghově prostoru odpovídají přímkám procházejícím stejným bodem v obrázku.



Obrázek 3.1: Houghova transformace pro detekci přímek [27]



(a) Parametrizace přímky (b) Parametrizace elipsy

Obrázek 3.2: Parametrizace objektů

Algoritmus

Detekce objektů v obrázku začíná detekováním hran ve vstupním obrázku. K tomu využijeme *detektory založené na první derivaci*[1], případně komplexnější *Cannyho hranový detektor*[2].

Následně algoritmus postupně prochází všechny body a v případě nalezení nenulového pixelu se hledají přímky, ke kterým bod náleží. U detekovaných přímek vypočteme vzdálenost od počátku V a úhel v radiánech U (viz obrázek 3.2a). Tyto hodnoty použijeme v *akumulačním poli*, kde na pozici $[V, U]$ přičteme jedničku. Po průchodu celým obrázkem nalezneme v akumulačním poli nejvyšší hodnoty (dle stanoveného počtu detekovaných přímek) a následně vykreslíme přímky s danými úhly a vzdálenostmi.

Pro elipsu bude proces dost podobný, lišit se bude jen parametrizace. Elipsa je dána pěti parametry – středem (x_0, y_0), délkou hlavní poloosy a, vedlejší poloosy b a orientací α. Pro nalezení parametrů, a tím i jednoznačně určené elipsy, však stačí pouze tři body, v obrázku 3.2b jsou označeny jako $(x_1, y_1), (x_2, y_2)$ a (x, y) [28].

Jednotlivé kroky algoritmu detekce elips jsou uvedeny v algoritmu 1. Proměnné odpovídají parametrům z obrázku 3.2b, případně se nacházejí v popisu v předchozím odstavci.

Využití při klasifikaci

Výše uvedený algoritmus jsme opět implementovali v Pythonu, tentokrát s využitím knihovny *scikit-image*, která má v sobě mimo jiné zabudovanou funkci Houghovy transformace. Do programu jsme vložili 280 obrázků s parazitem a 300 obrázků bez parazita (získané jako true positive, resp. false positive z blob detektoru). Jako výstup jsme obdrželi původní obrázek s vykreslenou elipsou, kterou

[1]Detektory založené na první derivaci využívají toho, že hrany se nacházejí tam, kde má funkce v první derivaci lokální maximum. K detekování využívají masky (operátory), které využijeme k výpočtu konvoluce s funkcí obrázku.

[2]Cannyho hranový detektor je také založen na první derivaci, ale využívá i gaussovský filtr na vyhlazování a detekční operátor.

jsme získali jako výsledek Houghovy transformace. V případě správné klasifikace jsme dostali původní obrázek s vykreslenou elipsou kolem kleštíka.

Po kontrole výstupů však bylo zjevné, že ve velkém množství případů není Houghova transformace schopna klasifikovat kleštíky správně. V momentě, kdy se na obrázku vyskytovalo více objektů, byla elipsa vykreslena s naprosto odlišnými parametry. Naopak na některých obrázcích byl jako kleštík klasifikován i jiný objekt.

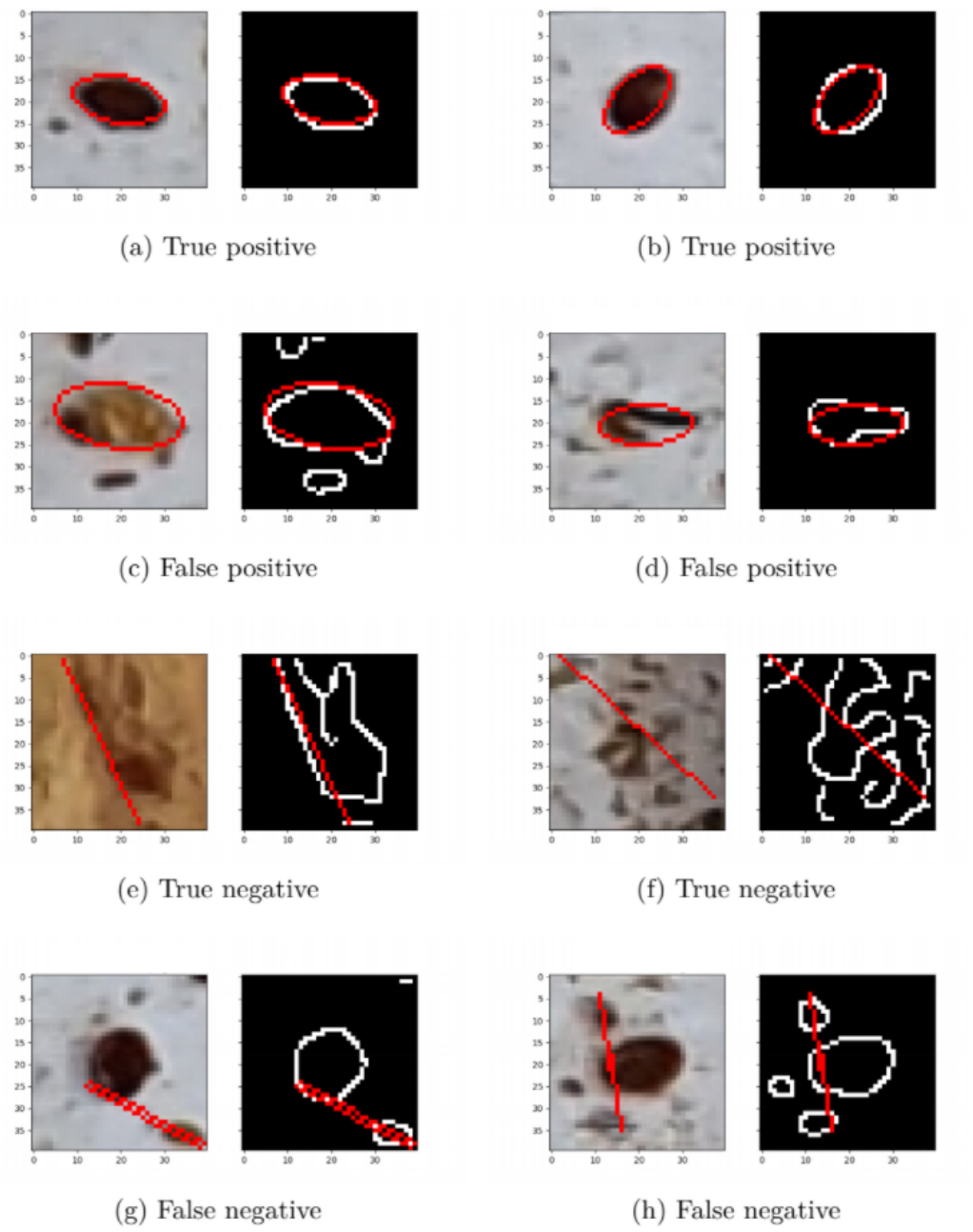
Abychom úspěšnost nějak vyčíslili, určili jsme velikosti čtyř základních skupin – true positive, false positive, true negative a false negative (viz obrázek 3.3). Hodnoty jsme následně dosadili do vzorců pro *accuracy*, *recall* a *precision*, tedy do parametrů určujících úspěšnost detekce s následujícími předpisy:

$$
\begin{aligned}
accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\
recall &= \frac{TP}{TP + FN} \\
precision &= \frac{TP}{TP + FP}
\end{aligned}
$$

Po dosazení jsme zjistili, že při využití našeho data setu dosahuje klasifikátor accuracy 75,38 %, precision 78,66 % a recall 68,62 %. Vzhledem k poměrně nízké úspěšnosti klasifikace jsme klasifikátor nepodrobili větší sadě obrázků a rovnou usoudili, že pro naše potřeby není vhodný. Proto se rovnou přesuňme k další kapitole, kde bude představena jiná metoda.

Algorithm 1: Houghova transformace pro detekci elips

Ulož všechny pixely hran do pole;
foreach *bod* (x_1, y_1) **do**
  foreach *bod* $(x_2, y_2) \neq (x_1, y_1)$ **do**
    d_1 = vzdálenost mezi (x_1, y_1) a (x_2, y_2);
    if $d_1 > min_{d_1}$ **then**
      $x_0 = (x_1 + x_2)$ / 2;
      $y_0 = (y_1 + y_2)$ / 2;
      $a = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ / 2;
      $\alpha = \arctan\,[(y_1 - y_2)$ / $(x_1 - x_2)]$;
    end
    foreach *bod* $(x, y) \neq (x_2, y_2)$ **do**
      d = vzdálenost mezi (x_0,y_0) a (x, y);
      if $d > min_d$ **then**
        $\beta = \arccos\,[(a^2 + d^2 - f^2)$ / $(2ad)]$;
        $b = \sqrt{(a^2 d^2 \sin^2 \beta)$ / $(a^2 - d^2 \cos^2 \beta)}$;
        Přičti jedničku v akumulačním poli na pozici b;
      end
    end
    Najdi maximální hodnotu v Houghově prostoru;
    if *b s maximálním výskytem v Houghově prostoru je přípustné* **then**
      Ulož spočtené parametry $(x_0, y_0), \alpha, a, b$;
      Vykresli elipsu na vložený obrázek;
      Vymaž body odpovídající detekované elipse z pole hran;
      Vymaž Houghův prostor;
    end
  end
end



(a) True positive (b) True positive

(c) False positive (d) False positive

(e) True negative (f) True negative

(g) False negative (h) False negative

Obrázek 3.3: Výsledky klasifikace Houghovou transformací (bílá čára odpovídá detekované hraně, červená detekované elipse). Vlevo je zobrazen výřez snímku, který jsme dostali na výstupu blob detektoru. Vpravo je výstup hranového detektoru s Houghovou transformací.

4. Detekce konvoluční neuronovou sítí

Druhý způsob detekce využívá konvoluční neuronovou síť. Tento přístup je sofistikovanější, umožňuje přesnější detekce. Jeho přesnost vychází z toho, že již nemusíme tělo kleštíka včelího aproximovat elipsou, ale můžeme pracovat s konkrétní podobou parazita. To zaručuje odhalení i takového jedince, jehož tělo se parametricky odlišuje od kuželosečky. Zároveň konvoluční neuronová síť produkuje výrazně menší množství falešně pozitivních výsledků detekce.

V první části kapitoly vysvětlíme celý koncept neuronových sítí a postup při trénování dat. Následně se přesuneme ke konvolučním neuronovým sítím a k popisu vyhodnocování úspěšnosti trénování.

4.1 Úvod do neuronových sítí

Pro pochopení toho, co jsou neuronové sítě, si nejprve stručně vysvětleme základní pojmy s nimi spojené.

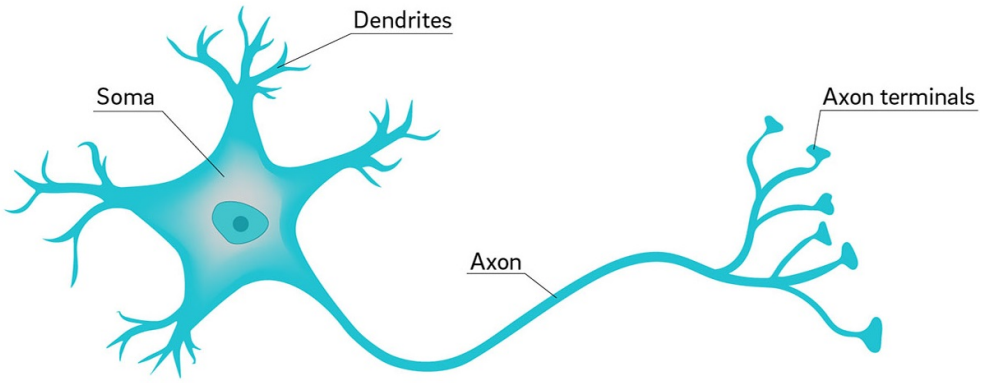
- **Umělá inteligence**: jakákoliv technika, která umožňuje počítači napodobit lidské chování.
- **Strojové učení**: umělá inteligence, která je schopna se učit bez explicitního naprogramování.
- **Hluboké učení**: strojové učení, které ke své činnosti využívá neuronové sítě. Je schopné řešit komplexnější problémy.

Po zbytek kapitoly se budeme věnovat hlubokému učení, přesněji neuronovým sítím. Nejprve si popíšeme, jak taková síť vypadá, z čeho se skládá a jak probíhá trénování dat. Poté se zaměříme na konvoluční neuronovou síť, která nám poslouží pro detekci kleštíka včelího.

Neuronová síť se skládá z velkého množství propojených neuronů, jejichž struktura a funkčnost se podobají neuronům v nervové tkáni člověka. Ve srovnání s jinými metodami strojového učení mají neuronové sítě adaptivní přístup k řešení různých problémů. Běžné algoritmy se obvykle zaměřují na řešení konkrétních problémů pomocí jednotlivých příkazů. Naproti tomu neuronové sítě nespoléhají na složitý systém zpracování, ale na vzájemně propojené neurony, které pracují paralelně. Postupným učením z vložených trénovacích dat se síť zdokonaluje a přizpůsobuje, stejně jako při procesu učení člověka [29].

Nervový systém v lidském těle obsahuje řadu specializovaných neuronů, které spolu komunikují prostřednictvím složitých spojení. Tento proces se nazývá *neurotransmise* a spočívá v přenosu informace pomocí elektrických a chemických signálů. Neurony jsou vzájemně propojeny *dendrity*, jež slouží jako receptory naslouchající příchozím signálům z jiných neuronů. V závislosti na vstupním signálu pak mohou buňky vyslat impuls přes svůj axon k přenosu nového signálu pomocí *presynaptických terminálů*.

Struktura umělé neuronové sítě je založena na stejném principu. Nejprve přijme informaci, následně ji zpracuje a na závěr vygeneruje výstup. V následujících podkapitolách si schéma neuronové sítě a fungování neuronů popíšeme detailněji.

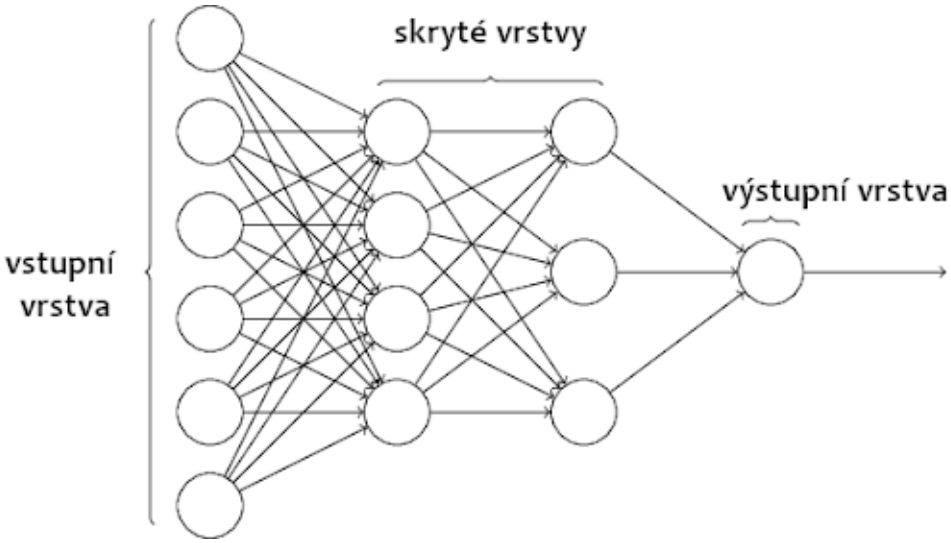


Obrázek 4.1: Stavba neuronu v lidském těle [30]

Schéma neuronových sítí

Neuronové sítě se skládají z řady vzájemně propojených neuronů, které jsou uspořádány do tří typů vrstev [31]. První z nich je *vstupní vrstva*, jež je zodpovědná za příjem vstupů a slouží jako rozhraní mezi daty a sítí. Velkým benefitem hlubokého učení jsou *skryté vrstvy*, které běžné algoritmy strojového učení neobsahují. Těch může být více a v závislosti na jejich množství roste jak časová složitost, tak i množství řešitelných problémů. Poslední vrstvou je *výstupní vrstva* představující výsledek celého procesu.

V závislosti na propojení mezi neurony ve skryté vrstvě rozlišujeme dvě základní kategorie neuronových sítí – dopřednou a rekurentní [29]. V první z nich se přenáší informace pouze jedním směrem. Celý proces je tedy ukončen při předání výstupu neuronu v poslední vrstvě. Rekurentní síť obsahuje neurony, které mohou být propojeny s jakýmkoliv neuronem ve skryté vrstvě bez ohledu na pořadí vrstev. Rovněž se v ní mohou nacházet smyčky.

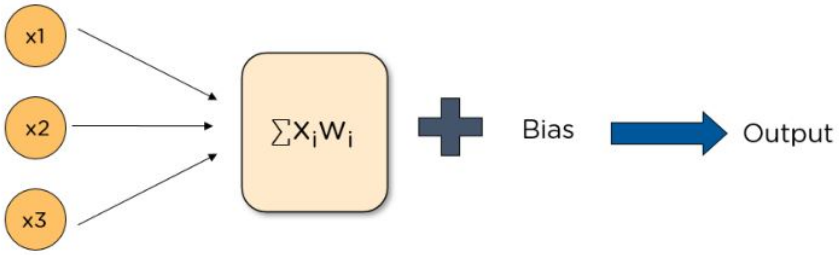


Obrázek 4.2: Vrstvy v neuronových sítích [32]

Aktivační funkce

V každém neuronu se provádějí následující operace (schéma viz obrázek 4.3):

- Každou vstupní hodnotu vynásobíme s váhou kanálu.
- Hodnoty z předchozího kroku sečteme.
- K této sumě přičteme *bias*.
- Konečný součet vložíme do určité *aktivační funkce*



Obrázek 4.3: Schéma operací v neuronu [33]

Aktivačních funkcí je celá řada. Ty nejvyužívanější jsou uvedeny níže, včetně jejich grafů (obrázek 4.4).

První funkcí je **prahová funkce**, jež je zadána předpisem

$$P(x) = \begin{cases} 0 & \text{pokud } x < n \\ 1 & \text{pokud } x \geq n. \end{cases}$$

Běžně se volí $n = 0$. Tato funkce se však příliš neuplatňuje, neboť není možné ji využít při procesu zpětného šíření.

Další možností je využití **sigmoidu** σ. Jeho výhodou je, že pro malé změny vah a biasu způsobí nepatrné změny výstupu. Naopak nevýhodou je *problém mizejícího gradientu*, což v konečném důsledku může znemožnit úspěšné trénování dat. Vstupní hodnoty σ mohou být z celého intervalu [0,1]. Výstup se řídí tímto funkčním předpisem:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

Hyperbolický tangens se velmi podobá předchozí funkci, jen jeho obor hodnot je interval $[-1, 1]$. Funkce je definována předpisem

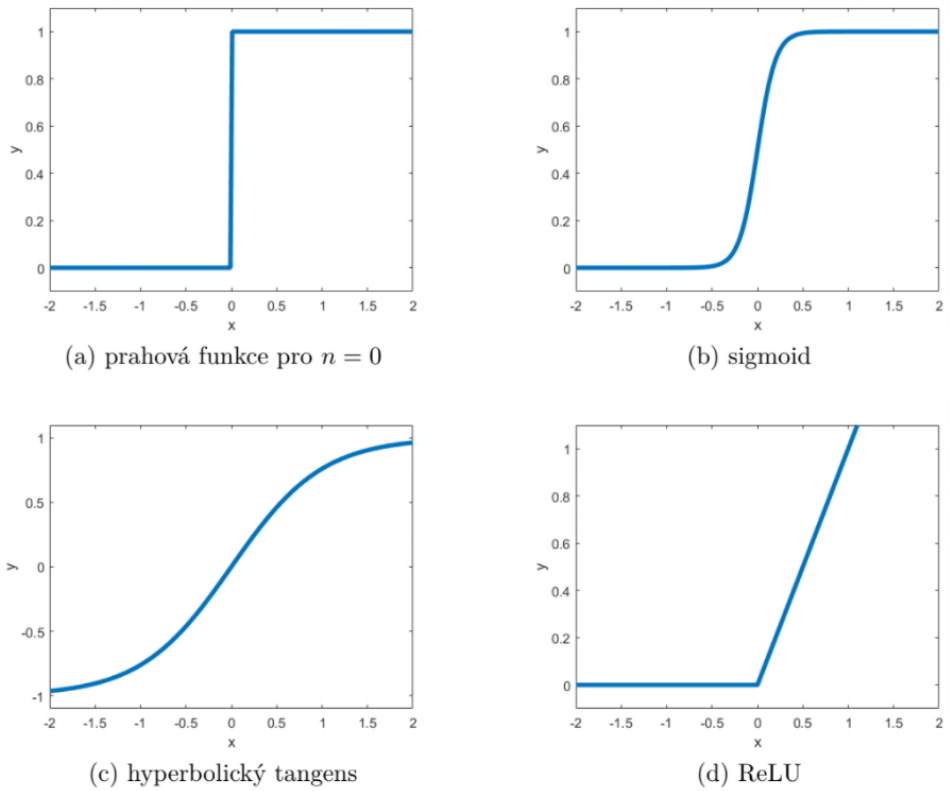
$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}.$$

Hyperbolický tangens do jisté míry eliminuje problém mizejícího gradientu, ale důsledkem je zpomalení celého trénování neuronové sítě.

Hojně využívanou funkcí je **Rectified Linear Units (ReLU)**, která je definována jako

$$ReLU(x) = max\{0,x\}.$$

Díky tomu, že její výpočet je nenáročný, představuje vhodnou alternativu sigmoidu či hyperbolického tangentu, zejména ve složitějších strukturách s miliony neuronů (například v konvolučních neuronových sítích).



(a) prahová funkce pro $n = 0$ (b) sigmoid

(c) hyperbolický tangens (d) ReLU

Obrázek 4.4: Grafy aktivačních funkcí

Trénování neuronových sítí

Před vysvětlením procesu trénování sítě si ještě definujme **ztrátovou funkci** L. Ta ukazuje, jak dobrá je naše neuronová síť pro určitý úkol, a je definována předpisem

$$L(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2.$$

y označuje číslo, které jsme chtěli získat, zatímco $\hat{y}$ představuje číslo, jež jsme dostali na výstupu. Sčítáme přes jednotlivé trénovací pokusy.

Trénování neuronových sítí začíná tím, že náhodně inicializujeme váhy. Lze očekávat, že hodnota ztrátové funkce bude zpočátku vysoká. Naším cílem je minimalizovat ztrátovou funkci, čímž získáme vyšší přesnost a kvalitnější síť.

Jednou z možností, jak ztrátovou funkci minimalizovat, je využití **gradientního sestupu** [31, 34, 35]. Pro definici gradientního sestupu označme ΔL jakožto malou změnu L vyvolanou vektorem

$$\Delta v = (\Delta v_1, \cdots, \Delta v_m)^T.$$

ΔL je přibližně rovna součinu gradientu ∇L s Δv. Pro připomenutí, gradient ∇L odpovídá vektoru

$$\nabla L = \left(\frac{\partial L}{\partial v_1}, \cdots, \frac{\partial L}{\partial v_m} \right)^T.$$

Výše uvedený vektor Δv můžeme zapsat jako

$$\Delta v = -lr \cdot \nabla L,$$

kde lr označuje *rychlost učení*. Tento parametr volíme tak, aby $\nabla L \cdot \Delta v$ co nejlépe aproximovalo ΔL. Opakovaným využitím následujícího vztahu snížíme gradient na minimum.

$$v \to v' = v - lr \cdot \nabla L$$

Při tréninku neuronové sítě hledáme váhy w_i a b_j minimalizující ztrátovou funkci L obdobně jako v předchozím odstavci:

$$\begin{aligned} w_i &\to w_i' = w_i - lr\frac{\partial L}{\partial w_i} \\ b_j &\to b_j' = b_j - lr\frac{\partial L}{\partial b_j}. \end{aligned}$$

Abychom mohli využít gradientního sestupu a mohli tak minimalizovat ztrátovou funkci L, potřebujeme znát postup, jak gradienty vypočítat. K tomu slouží **algoritmus zpětného šíření** [31]. Ten začíná ve výstupní vrstvě a zpětným chodem neuronovou sítí přepočítává v každé vrstvě váhy a bias.

Nejprve si zaveďme značení. Nechť N označuje poslední vrstvu, $N-1$ předposlední vrstvu (další vrstvy analogicky). Nechť f označuje aktivační funkci, $u^{(i)}$ neuron v i-té vrstvě a y předpokládanou výstupní hodnotu. Potom

$$\begin{aligned} z^{(N)} &= w^{(N)} \cdot a + b \\ u^{(N)} &= f(z^{(N)}) \\ L &= (a^{(N)} - y)^2 \end{aligned}$$

Hodit se nám bude rovněž *věta o derivaci složené funkce*, proto ji nyní zformulujme:

Věta 1. *Nechť funkce $g_k : \mathbb{R}^n \to \mathbb{R}$ mají v bodě a totální diferenciál a funkce $f : \mathbb{R}^k \to \mathbb{R}$ má v bodě $b = (g_1(a), \ldots, g_k(a))$ totální diferenciál. Definujme funkci $h(x) = f(g_1(x), \ldots g_k(x))$. Pak h má v a totální diferenciál a pro $i = 1, \ldots, n$ platí*

$$\frac{\partial h}{\partial x_i}(a) = \sum_{j=i}^{k} \frac{\partial f}{\partial y_j}(b)\frac{\partial g_j}{\partial x_i}(a).$$

Označme $P(u^i)$ předchůdce neuronu $u^{(i)}$. Potom s využitím poznatků z předchozích odstavců můžeme ukázat schéma algoritmu zpětného šíření.

Algoritmus nejprve spustí *Dopředné šíření*, které prochází sítí od prvního uzlu a v každém uzlu počítá funkční hodnotu aktivační funkce. Na výstupu vrací poslední neuron. Následně v něm spočítá pomocí Věty 1 jednotlivé parciální derivace a takto pokračuje i se zbylými neurony.

Algorithm 2: Dopředné šíření

for $i = 1, \dots n$ **do**
  $\mathbb{A}^{(i)} = \{u^{(j)} \mid j \in P(u^{(i)})\}$;
  $u^{(i)} = f^{(i)}(\mathbb{A}^{(i)})$
end
return $u^{(n)}$

Algorithm 3: Zpětné šíření

Spusť algoritmus 2;
$g^{(n)} = 1$;
for $i = n-1, \dots, 1$ **do**
  $g^{(i)} = \sum_{j:i\in P(i^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(j)}} \frac{\partial u^{(j)}}{\partial u^{(i)}}$
end
return g

4.2 Konvoluční neuronové sítě

Konvoluční neuronová síť vyžaduje menší množství trénovacích dat, neboť dokáže rozpoznávat jeden objekt na různých pozicích, v různých velikostech a v různých orientacích. Jiné typy neuronových sítí by detekovaly například kleštíka v dolním levém rohu pouze v případě, kdy by trénovací data obsahovala snímek s kleštíkem ve zmíněné pozici.

Toto vylepšení je založeno na využití stejných vah pro celý obrázek. Například při aplikaci filtru (sady vah) velikosti 3×3 potřebujeme pouze 9 vah a 1 bias pro každou příznakovou mapu, zatímco při využití klasické neuronové sítě bychom pro obrázek velikosti 30×30 s 20 neurony potřebovali více než 18 000 parametrů.

Konvoluční neuronové sítě obsahují *konvoluční vrstvy* a v menší míře také dříve popsané *fully-connected* vrstvy.

Konvoluční vrstvy

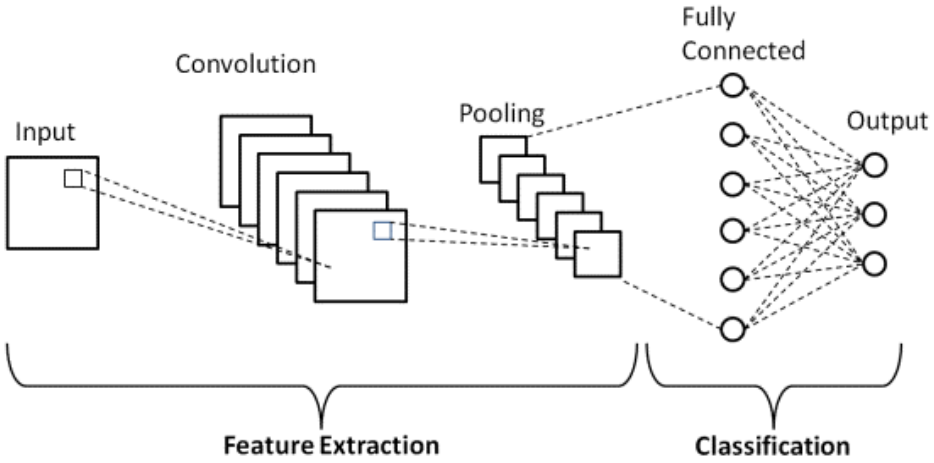
Než se podíváme na konvoluční vrstvy, definujeme si základní operaci, kterou konvoluční vrstvy využívají.

Definice 1. *Konvoluci* $*$ *definujeme jako zobrazení, které dvěma funkcím* f, g *přiřadí funkci definovanou předpisem*

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt.$$

Diskrétní konvoluce je pak definována vztahem

$$(K * I)_{i,j} = \sum_{m,n} I_{i-m,j-n} K_{m,n}.$$



Obrázek 4.5: Schéma konvoluční neuronové sítě [36]

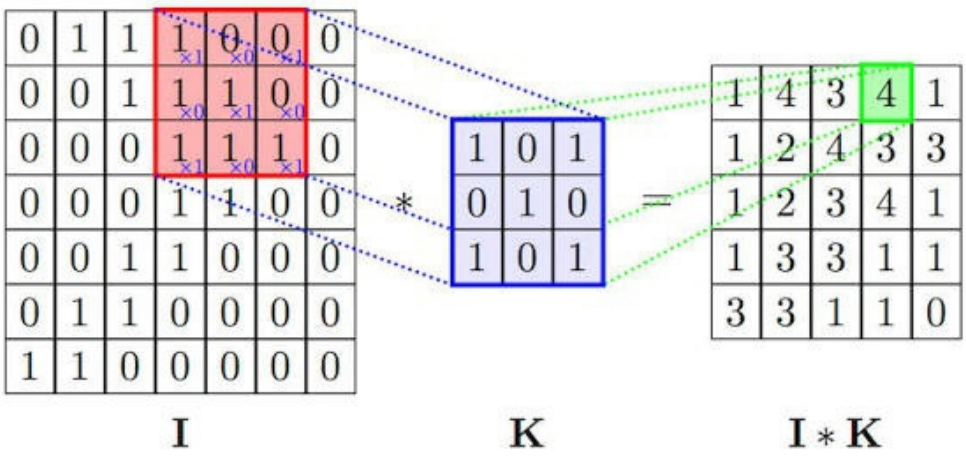
Podobně definujeme cross-korelaci, která se vypočte následovně:

$$(K \star I)_{i,j} = \sum_{m,n} I_{i+m,j+n} K_{m,n}.$$

V konvoluční vrstvě se na obrázek I použije filtr K. Každý obrázek se skládá z n kanálů (třeba v případě RGB obrázků jde o tři kanály), tedy jediný pixel je ve skutečnosti zastoupen n-dimenzionálním vektorem. Filtr obsahuje jinou sadu vah pro každou vstupní dimenzi [31, 37]. Vzhledem k tomu, že počet výstupních kanálů se může lišit, bývá filtr reprezentován 4-dimenzionálním tenzorem.

Nechť W představuje šířku filtru, H jeho výšku, F počet výstupních kanálů, C počet vstupních kanálů a S odpovídá velikosti „kroku“ (výstupní pixely se počítají pro každý S-tý vstupní pixel). Potom filtr má celkovou velikost $W \times H \times C \times F$ a výstup z konvoluce vypadá následovně:

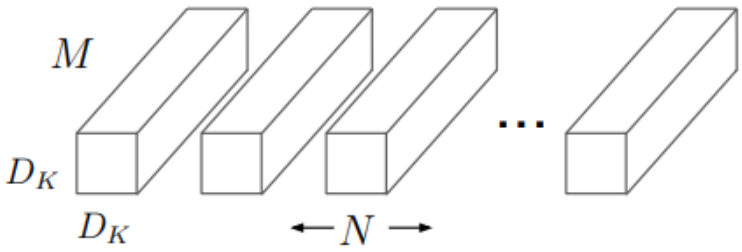
$$(K \star I)_{i,j,o} = \sum_{m,n,c} I_{i \cdot S+m, j \cdot S+n} K_{m,n,c,o}.$$



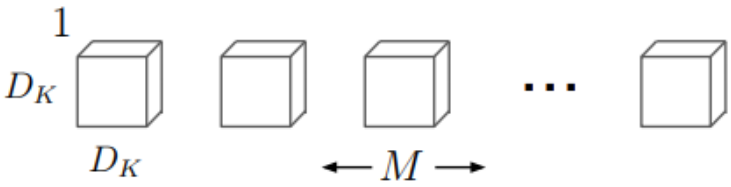
Obrázek 4.6: Konvoluce obrázku s filtrem [38]

V některých typech neuronových sítí se využívá „levnější“ **separable konvoluce** [31], která se skládá z

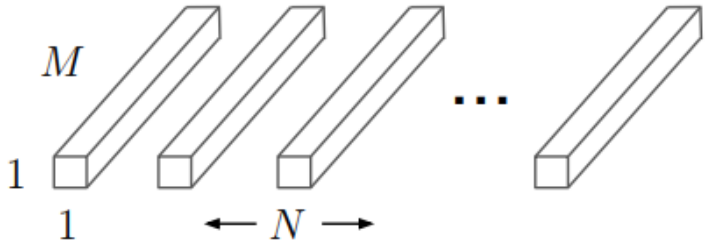
- **Depthwise separable** konvoluce, která se aplikuje na každý kanál zvlášť. Tím se redukuje časová a paměťová složitost. Nevýhodou je, že nebere v potaz ostatní kanály, proto však následuje druhá část.
- **Pointwise separable** konvoluce 1×1, která kombinuje výstupy z předchozí konvoluce a vygeneruje potřebný počet výstupních kanálů.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Obrázek 4.7: Standardní versus separable konvoluce [39]

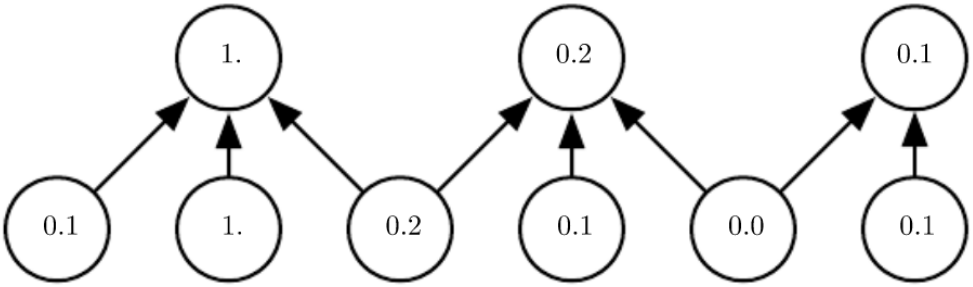
Po využití konvoluce přichází na řadu aktivační funkce, zpravidla ReLU. Před vstupem do nové vrstvy proběhne ještě *pooling*, jenž je popsán v následující sekci.

Pooling

Pooling představuje operaci, která obdobně jako konvoluce redukuje počet parametrů. Přitom se ale informace agregují do větších celků, což nám umožňuje s každým poolingem vědět o obrázku více.

Jako operace se volí maximum, průměr, případně $L2$ norma. V praxi to vypadá tak, že na obrázek postupně aplikujeme filtr malého rozměru a z dané oblasti

vybereme hodnotu odpovídající maximu (resp. průměru, $L2$ normě). Většinou se využívá maxima, neboť tato operace dodává neuronové síti translační invariantnost, tedy schopnost detekovat objekt i v případě, že se na obrázku nachází na jiném místě.



Obrázek 4.8: Max pooling [37]

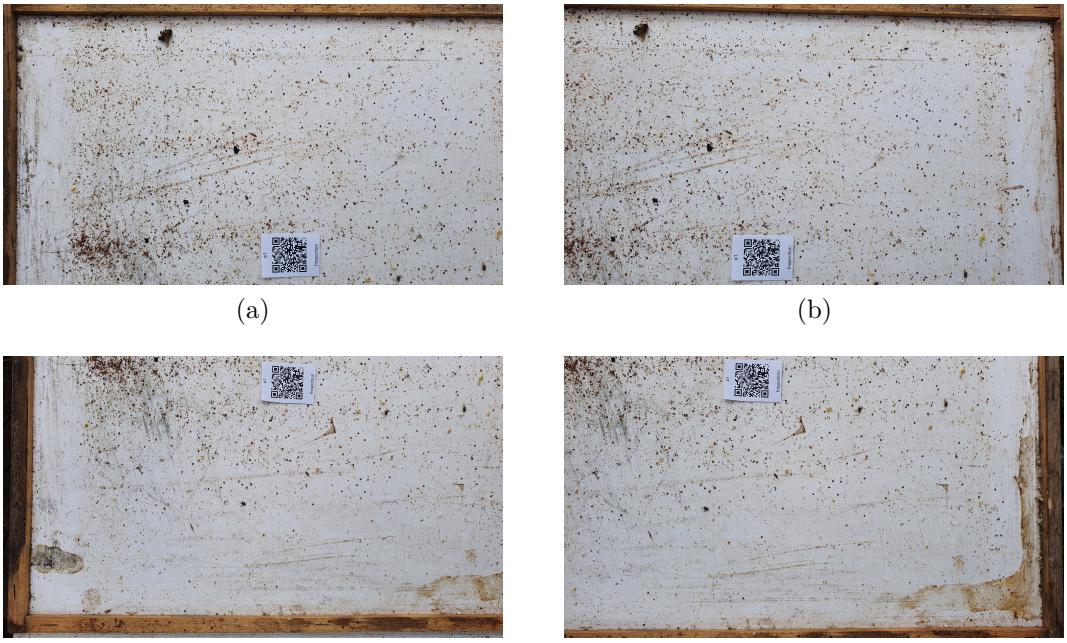
5. Aplikace na detekci kleštíka včelího

Poznatky z předešlých kapitol nyní aplikujeme na detekci kleštíka včelího. Jak bylo uvedeno v úvodu, začneme sešíváním snímků spadových podložek. Následně na těchto podložkách proběhne anotace dat. Data vložíme do konvoluční neuronové sítě a spustíme trénovací část. Do natrénované sítě vložíme výstupy z blob detektoru a síť poté klasifikuje, kde se nachází kleštík a kde ne. Na závěr vyhodnotíme úspěšnost sítě.

5.1 Sešívání snímků spadové podložky

V této sekci si představíme proces sešívání snímků na konkrétních obrázcích spadové podložky. Program je implementován v Pythonu za použití knihovny OpenCV, jež obsahuje velké množství funkcí pro zpracování obrazu.

Samotný program je postaven na funkci *createStitcher*, která v sobě zahrnuje všechny kroky sešívání snímku (načtení obrázků, detekce příznaků, hledání korespondencí a odhad transformace mezi obrázky). Pro lepší představu o tom, jak funkce pracuje, si na obrázku z úlu[1] ukážeme detekované klíčové body a nalezené korespondence. Předtím však ještě ukážeme vstupní data a výstupní sešitý obrázek spadové podložky, se kterou jsme pracovali v této sekci.



(a) (b) (c) (d)

Obrázek 5.1: Spadová podložka – vstup

[1] Spadová podložka obsahuje velké množství klíčových bodů, proto by vstupní obrázek nebyl příliš názorný.

Vstup

Pro sešití snímků nejprve potřebujeme vložit obrázky, které se překrývají. Je dobré si uvědomit, že čím větší překryv, tím větší přesnost sešití. Obrázků může být libovolný počet a mohou být v libovolném rozložení. V našem případě jsme využili čtyři obrázky v pomyslné mřížce 2×2 (viz obrázek 5.1).

Výstup

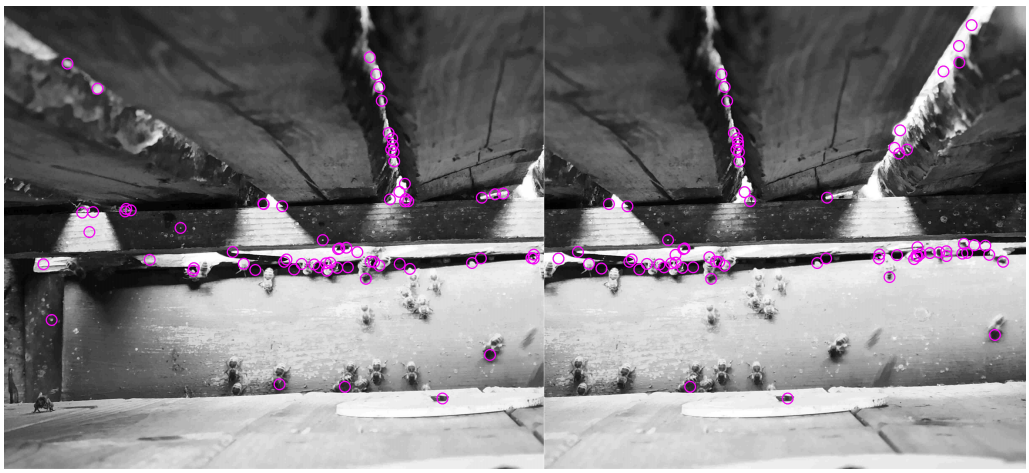
Výstupem je jeden sešitý obrázek. Ten má větší rozlišení, než jaké bychom dostali, kdybychom podložku zachytili na jediný záběr. Proto i detekce kleštíka vykazuje vyšší úspěšnost. Sešitému snímku mohou chybět některé body, některé naopak mohou být duplikované. To ovšem nemá na proces detekce kleštíka větší vliv, neboť množství roztočů na sešité podložce se tímto příliš nezmění.

Obrázek 5.2: Spadová podložka – výstup

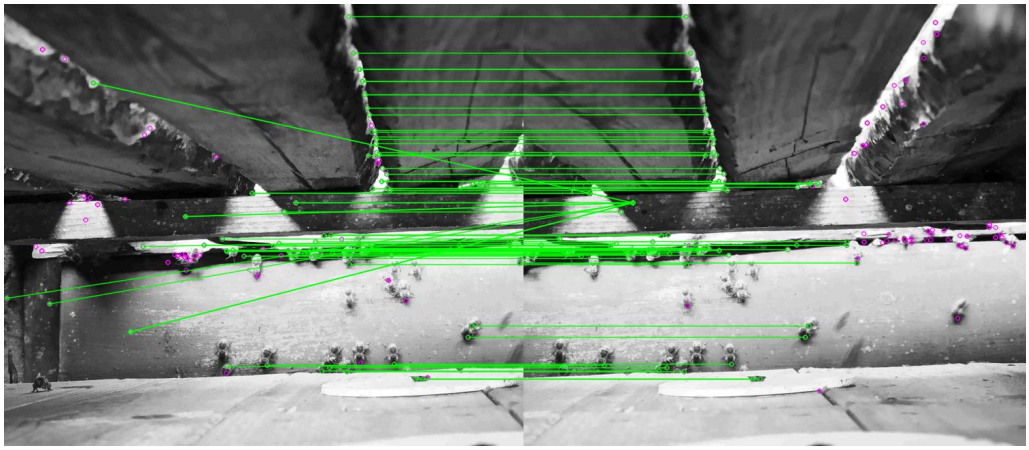
Detekce klíčových bodů a hledání korespondencí

V dalším kroku s využitím algoritmu SIFT detekujeme klíčové body a spočítáme jejich deskriptory. Na obrázku 5.3a jsou tyto body vyznačeny barevnými kružnicemi. Poté vyhledáme všechny možné korespondence nalezených klíčových bodů (viz obrázek 5.3b). Následně vypočteme matici transformace, k čemuž využijeme algoritmus RANSAC. V našem případě hledáme afinní transformaci. To vyplývá z charakteru našich sešívaných dat, ke kterým můžeme přistupovat jako k rovinné (planární) 2D scéně. Zároveň předpokládáme, že rovina senzoru fotoaparátu bude se spadovou podložkou téměř rovnoběžná. Korespondence, které splňují finální spočtenou transformaci, vidíme na obrázku 5.3c.

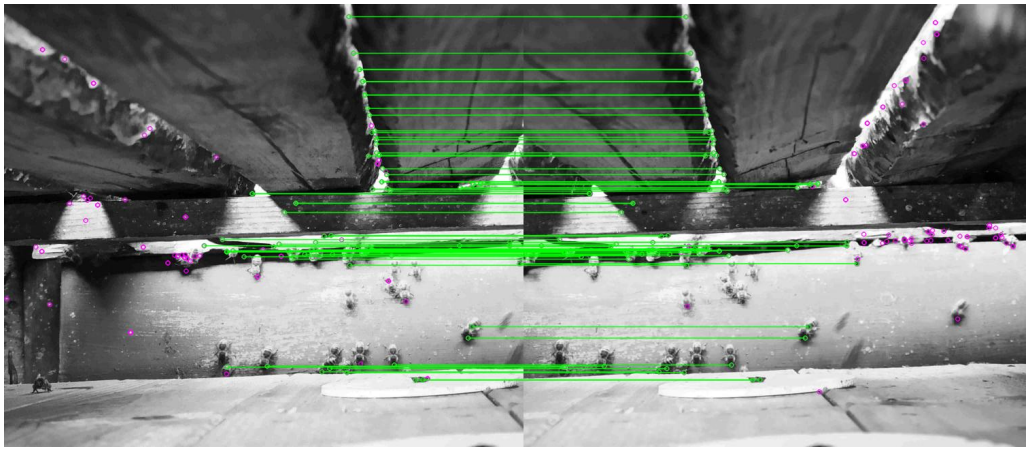
[1]Pro přehlednost vyznačeno pouze 150 klíčových bodů.



(a) Zobrazení klíčových bodů



(b) Zobrazení klíčových bodů s jejich korespondencemi společně s fialově vyznačenými outliery



(c) Korespondence klíčových bodů splňujících afinní transformaci

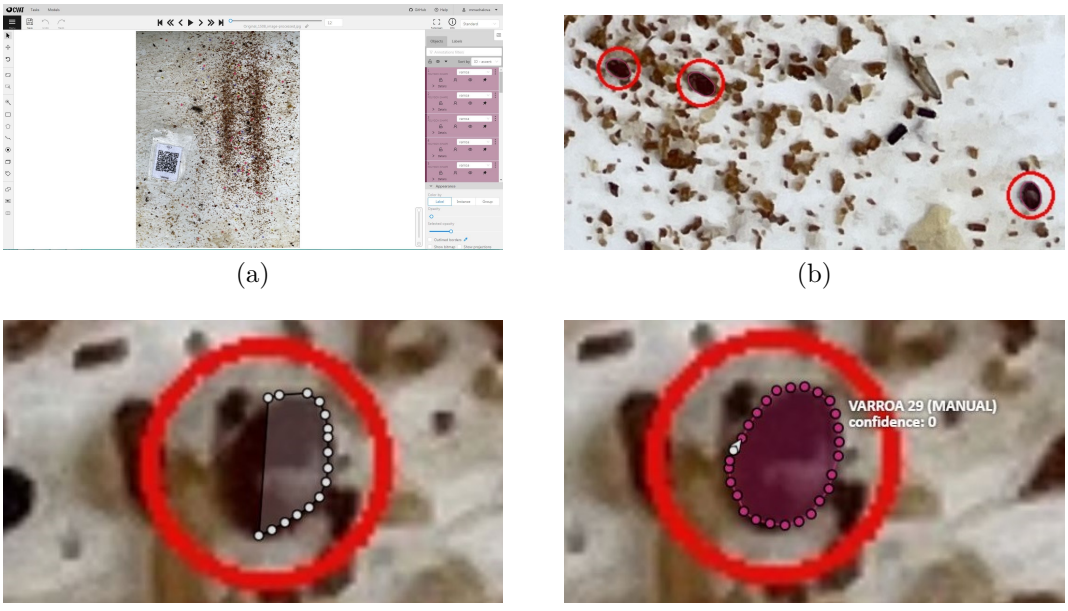
Obrázek 5.3: Zobrazení klíčových bodů a jejich korespondencí[2]

5.2 Anotace dat

Jak již bylo zmíněno, každá neuronová síť potřebuje velké množství dat, na základě kterých může zahájit proces učení a postupně se zlepšovat. Pro získání takových dat je potřeba provést *anotaci* (označení) dat. K tomu jsme využili *Computer Vision Annotation Tool*, volně dostupný webový nástroj pro anotaci obrázků a videí (vzhled pracovního prostředí je vyobrazen na obrázku 5.4a).

Proces anotace kleštíka včelího je vyobrazen na obrázcích 5.4c a 5.4d. Na každé spadové podložce bylo potřeba všechny roztoče ohraničit křivkou a následně uložit do databáze. Pro snadnější „ruční“ anotaci byla na některých podložkách provedena nejprve anotace „počítačová“ (na obrázcích 5.4 odpovídají tyto anotace červeným kružnicím). Ta však slouží pouze jako pomocný nástroj, všechny objekty je potřeba anotovat ručně. Vzhledem k enormnímu množství trénovacích dat se na anotaci kleštíka včelího podílelo přibližně dvacet lidí v rámci zmíněného projektu a společně jsme označili 8 125 roztočů.

Původně jsme zamýšleli využít některou segmentační neuronovou síť (např. U-Net), která pro své učení potřebuje i binární masku segmentovaného objektu. Vzhledem k vysoké úspěšnosti klasifikační sítě oproti síti segmentační bylo nakonec zvoleno řešení se sítí, jež pro své učení potřebuje jen výřezy s daným objektem bez těchto masek (toto řešení je popsáno níže).



(a) (b)

(c) (d)

Obrázek 5.4: Anotace dat

5.3 Blob detektor

V kapitole zabývající se detekcí pomocí klasických metod jsme popsali *blob detektor*. Ten se však ukázal užitečným i při využití konvolučních neuronových sítí, neboť jsme díky němu vytvořili sadu klasifikačních dat.

Z obrázku sešité spadové podložky jsme segmentovali obrázky velikosti 64×64 pixelů, a to tak, aby ve středu tohoto čtverce byl „kandidát“ na kleštíka. Pro každý detekovaný „blob“ jsme takto vytvořili a následně uložili jeden obrázek. Poté následovalo manuální třídění na obrázky s kleštíky a na obrázky, na kterých je vyobrazeno něco jiného. Na základě tohoto třídění jsme snímky rozdělili do kategorií *true positive* a *false positive* (viz obrázky 5.5 a 5.6). Detektor jsme měli nastavený tak, aby zachytil všechny roztoče na podložce. Vzhledem k citlivosti tedy na výstupu obdrželi i velké množství false positive. To však v konečném důsledku nepředstavuje problém, neboť klasifikace je následně s velkou úspěšností vytřídí.

Obrázek 5.5: True positive

Obrázek 5.6: False positive

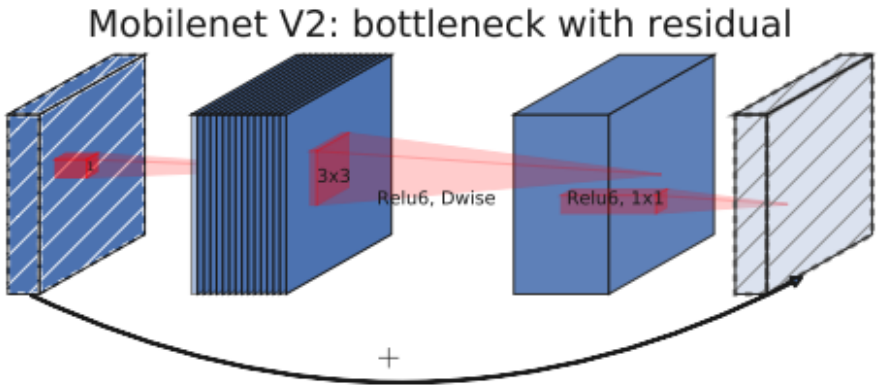
5.4 Mobilenet

Společnost Google vytvořila v roce 2017 první verzi Mobilenet, architektury uplatněné při klasifikaci kleštíka včelího. Cílem bylo vytvořit síť, která bude vyžadovat menší počet parametrů a zároveň menší komplexnost [40]. Během následujících dvou let byly vytvořeny další dvě verze a nejnovější z nich (*MobileNetV3-Small*) jsme použili v naší práci.

Architektura sítě

Pro popsání architektury MobilenetV3 si nejprve ukažme schéma předchozí verze MobilenetV2.

Základní komponentou jsou **invertní reziduální bloky**, které propojují vstupy do první konvoluce bloku s výstupy z poslední konvoluce. Tyto hodnoty se sečtou, což dává síti přístup k dřívějším aktivacím, které nebyly upravovány konvolučním blokem. Zatímco klasické reziduální bloky mají na vstupu velké množství kanálů, postupnými konvolucemi parametrů ubývá a až na závěr se počet kanálů znovu zvýší, u invertních reziduálních bloků je proces opačný, tedy počet parametrů roste a pak klesá (viz obrázek 5.7). To odpovídá tomu, že místo klasické konvoluce využijeme separabilní konvoluci. První a poslední vrstvě se v tomto schématu říká **bottleneck** (*„hrdlo láhve“*).



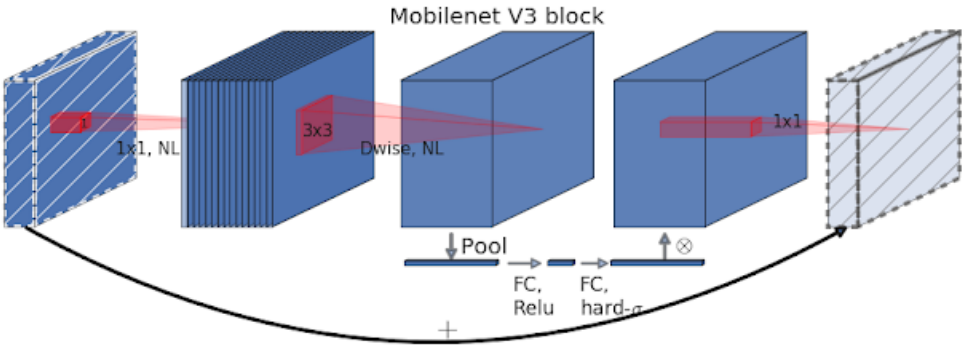
Obrázek 5.7: Schéma bloku MobileNetV2 [40]

MobilenetV3 vznikl tím, že byl přidán **Squeeze and Excitation Block**, tedy blok s následujícími operacemi:

- *squeeze*: výpočet průměrné hodnoty každého kanálu (globální pooling)
- *excitation*: výpočet vah všech kanálů využitím sigmoidu a následné vynásobení s odpovídajícím kanálem

Tento blok umožňuje globální interakci mezi kanály. Znamená to zároveň vyšší počet parametrů, ale přidáním skryté vrstvy s menším počtem parametrů dokážeme tento nárůst zmírnit.

Nyní si podrobněji popíšeme jednotlivé přípravné kroky s neuronovou sítí, která bude schopna s co největší přesností klasifikovat kleštíky včelí.



Obrázek 5.8: Schéma bloku MobileNetV3 [40]

Příprava dat

Jako vstupní obrázky jsme použili výřezy velikosti 64×64 pixelů. Obrázky byly rozděleny do dvou sad (skupin). První sada obsahovala 8 125 obrázků s kleštíky, jež vznikly na základě anotací. Druhá sada, tvořená 50 000 obrázky, obsahovala objekty na náhodných místech, kde nebyly anotace. Následně anotátoři zkontrolovali, že tyto obrázky skutečně kleštíka neobsahují.

Trénovací data byla augmentována v každé epoše náhodnými transformacemi. Obrázky byly se zadanou pravděpodobností zrcadlově převráceny či rotovány. Náhodně jim byl přidáván šum a pozměněn kontrast. Díky této transformaci dokážeme podchytit větší množství rozdílných „pozic“ roztoče na spadové podložce a je také možné zvýšit pravděpodobnost správné detekce na rozmazané fotografii.

Trénování

Trénování sítě proběhlo během 500 epoch. Při každé epoše bylo vybráno 512 obrázků, které byly vloženy do sítě s výše popsanou architekturou. Z vybraných obrázků bylo ještě odděleno 10 % obrázků pro validační část, zbylé obrázky se podrobily trénování. Na závěr každé epochy bylo provedeno vyhodnocení úspěšnosti modelu během tohoto cyklu. Konkrétně jsme počítali *accuracy, F1 score* a ztrátovou funkci (*loss*). První dvě zmíněné veličiny mají následující předpis:

$$\begin{aligned} \text{accuracy} &= \frac{TP + TN}{TP + FP + TN + FN} \\ \text{F1 score} &= 2 \cdot \frac{precision \cdot recall}{precision + recall}. \end{aligned}$$

*F*1 *score* představuje harmonický průměr recall a precision, čímž eliminuje extrémy z naměřených hodnot. Funkce loss může nabývat různých podob. V našem případě se jedná o kombinaci *Softmaxu* a *Negative Log-Likelihood* (NLL), pro které platí:

$$\begin{aligned} \text{Softmax}(\mathbf{x})_\text{i} &= \frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_j)} \\ \text{NLL}(\mathbf{x}) &= -\log(\mathbf{x}) \end{aligned}$$

kde $\mathbf{x} = (x_1, x_2, \ldots, x_K) \in \mathbb{R}^K$. Tedy pro třídu (class id) $i \in \{0, 1\}$ dostáváme výsledný vzorec pro výpočet loss:

$$\text{loss}(\mathbf{x}, \text{i}) = -\log\left(\frac{\exp(x_i)}{\sum_{j=0}^{1}\exp(x_j)}\right).$$

Validace

Po trénování následovala validace s obrázky, které jsme k tomu vyčlenili při přípravě dat. Validace slouží k posuzování kvality klasifikace. Jinými slovy, validace počítá predikce, které se porovnávají se skutečnou hodnotou výstupu. Pokud se tyto hodnoty shodují, znamená to, že síť je schopna předvídat výstupy správně i pro data, která neproběhla sítí během trénovací části.

I pro validaci jsme počítali accuracy, F1 score a loss a spolu s výsledky trénování jsme je zanesli do grafu (viz sekce 5.5). Poté přišla na řadu poslední část, tedy klasifikace.

Klasifikace

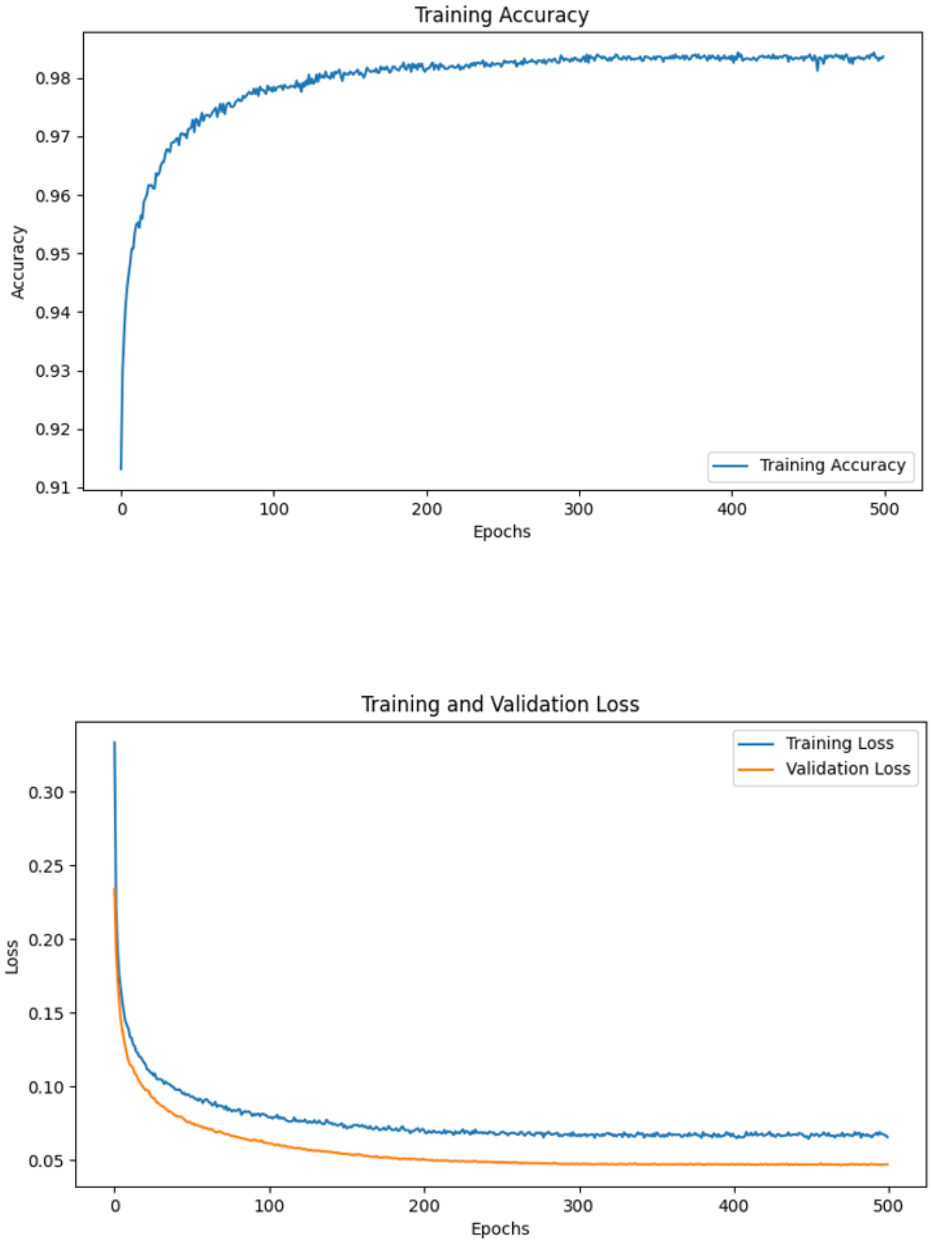
Po té, co jsme síť naučili rozpoznávat kleštíky, jsme mohli postoupit ke klasifikační části. Na sešité spadové podložce jsme nechali detekovat kandidáty na kleštíky, které jsme poté vložili do naučené neuronové sítě. Ta nám vyhodnotila, zda se na obrázku nachází zástupce kleštíka včelího, nebo ne.

Nyní se přesuňme k výsledkům trénování, validace a klasifikace.

5.5 Výsledky

Výsledky trénování a validace

Jak jsme již zmínili, trénování a validace proběhly v 500 epochách. Už po první epoše měla naše síť přesnost 91,31 %. Validační loss začínal na hodnotě 23,39 % a trénovací na 33,35 %, což není příliš příznivé. Avšak už v průběhu deseti epoch klesly tyto hodnoty pod 10 %. Po ukončení 500. epochy síť dosáhla train accuracy **98**,**37** %, train loss **6**,**57** % a validation loss **4**,**71** %.



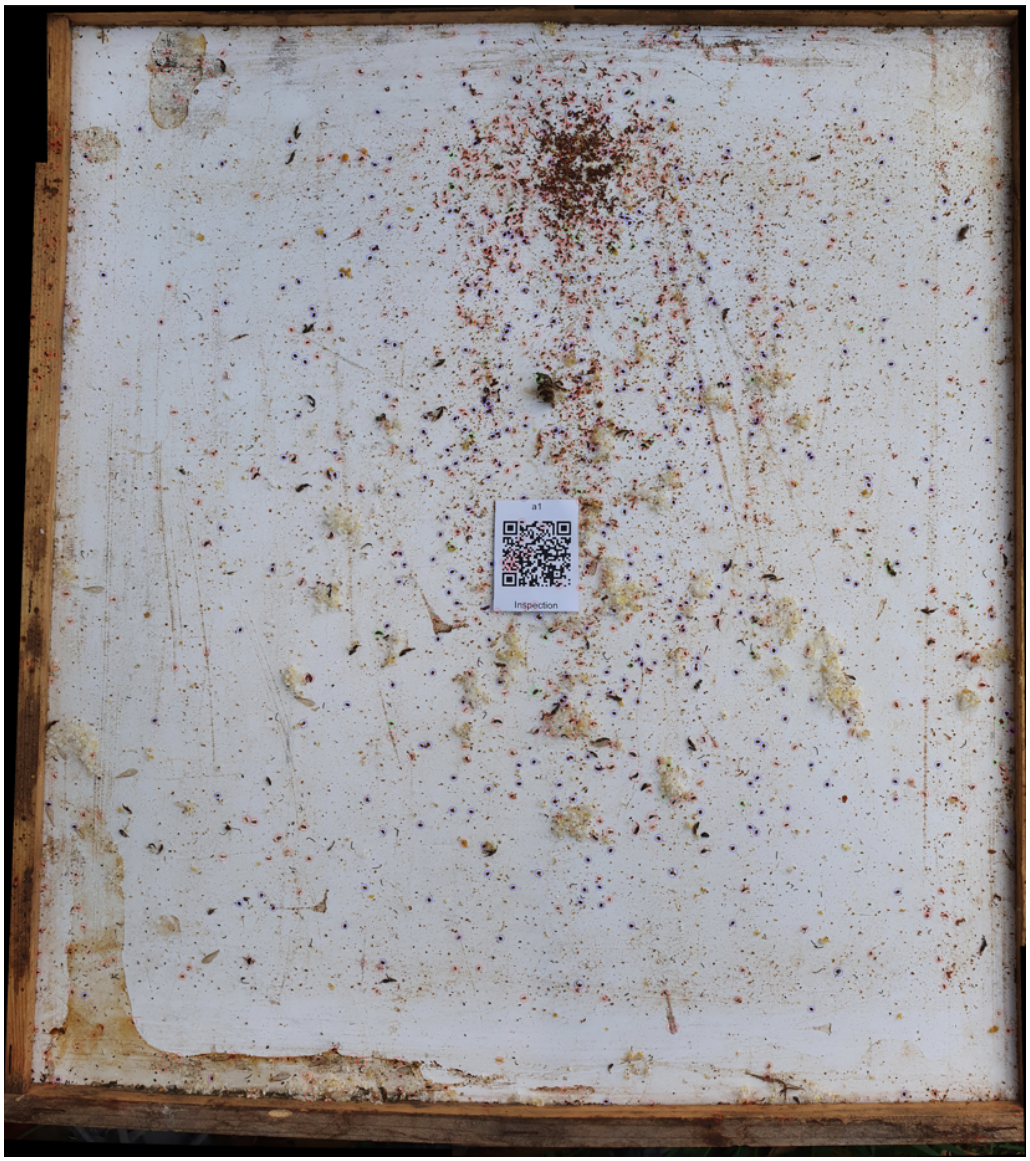
Obrázek 5.9: Výsledky trénování a validace

Výsledky klasifikace

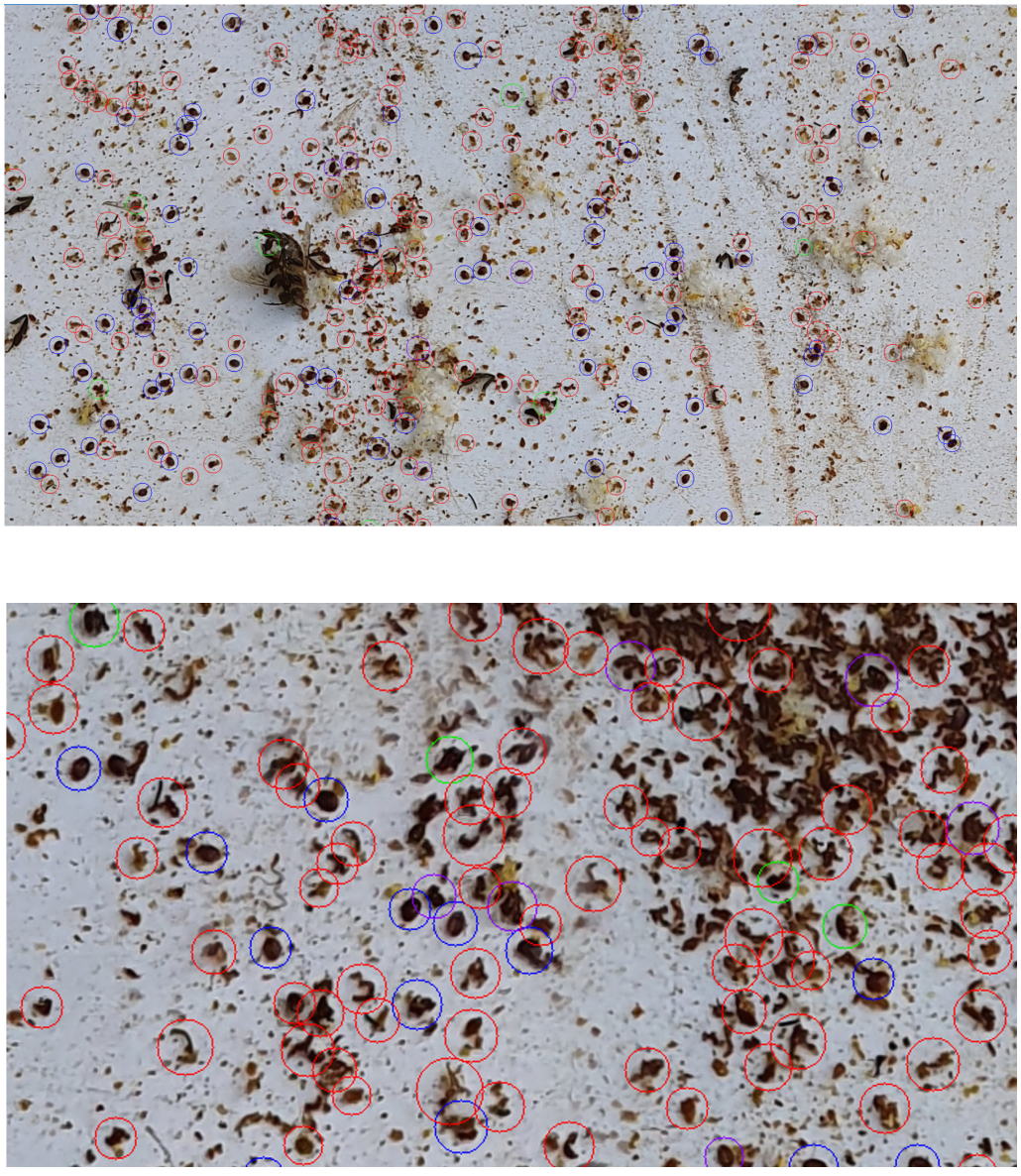
Výslednou klasifikaci jsme se rozhodli znázornit na obrázku vložené spadové podložky. Na něj jsme vykreslili kružnice kolem objektů detekovaných blob detektorem. K vykreslení jsme využili více barev, přičemž rozdělení barev dle výsledku klasifikace je následující:

- červená – zamítnuto, pravděpodobnost správné klasifikace ≥ 70 %
- zelená – zamítnuto, pravděpodobnost správné klasifikace < 70 %
- fialová – potvrzeno, pravděpodobnost správné klasifikace < 70 %
- modrá – potvrzeno, pravděpodobnost správné klasifikace ≥ 70 %

Z obrázků 5.10 a 5.11 je patrné, že klasifikací jsme vyřadili velké množství false positive.



Obrázek 5.10: Výsledky klasifikace výstupů z blob detektoru



Obrázek 5.11: Výsledky klasifikace výstupů z blob detektoru – přiblíženo

Závěr

Cílem této práce bylo prozkoumat možnosti detekce kleštíka včelího pomocí počítačového vidění. Na úvod jsme se seznámili s problematikou varroázy v České republice a s dosavadními postupy při monitoringu včelstva.

Následně jsme se zaměřili na sešívání snímků. Rozebrali jsme si jednotlivé kroky a zaměřili se především na detekci příznaků a hledání korespondencí. Popsali jsme teoreticky i matematicky Harrisův detektor rohů a dvě nejužívanější metody detekce – SIFT a SURF, jejichž koncept vychází z tohoto detektoru. Na závěr kapitoly jsme se seznámili s RANSACem, algoritmem pro odhad transformace mezi sešívanými obrázky.

Třetí kapitola byla zaměřena na detekci klasickými metodami, bez využití neuronových sítí či obdobných pokročilých struktur. Vyzkoušeli jsme blob detektor, který převede obrázek do binárního obrázku a následně hledá útvary dle zadaných parametrů (obsah, kruhovitost, konvexita, poloměr setrvačnosti). Výstupy jsme klasifikovali pomocí Houghovy transformace pro detekci elips, avšak výsledek klasifikace dosahoval accuracy 75,38 %, což pro naše potřeby nebylo postačující. Proto jsme se rovnou přesunuli k detekci s využitím konvoluční neuronové sítě.

Abychom mohli vytvořit detektor s využitím neuronové sítě, bylo potřeba se seznámit se strukturou a fungováním neuronových sítí. Popsali jsme si operace prováděné v neuronu a funkce k tomu potřebné. Následně jsme se zaměřili na matematický popis trénování neuronových sítí a počítání ztrátové funkce. Poté jsme se přesunuli ke konvolučním neuronovým sítím. Popsali jsme konvoluční vrstvy a operace prováděné v těchto sítích.

V poslední kapitole jsme se dostali k implementaci konkrétní konvoluční neuronové sítě, kterou jsme vytvořili pro detekci kleštíka včelího. Začali jsme s přípravou trénovacích, validačních a klasifikačních sad. Naše data představovaly segmenty obrázků spadových podložek v úlu. Ty byly zachyceny na více fotografiích a následně sešity pomocí metod uvedených v druhé kapitole. Na těchto obrázcích spadových podložek jsme pomocí softwaru Computer Vision Annotation Tool anotovali kleštíky, čímž jsme získali potřebnou trénovací a validační sadu. Klasifikační sada byla vytvořena pomocí blob detektoru popsaného ve třetí kapitole.

Po získání všech potřebných dat jsme se mohli přesunout k implementaci konvoluční sítě. Využili jsme model MobilenetV3, který je využíván primárně pro práci s mobilními a vestavěnými zařízeními. Popsali jsme, jak v tomto modelu fungují invertní reziduální bloky, jakou roli hrají Squeeze and Excitation bloky a jak vypadá bottleneck.

Po teoretickém popisu jsme se zaměřili na jednotlivé kroky potřebné k natrénování sítě. Trénovací proces jsme spustili na CPU s využitím knihovny *PyTorch*. V rámci každé epochy jsme trénovací data podrobili náhodné transformaci pro zajištění větší nezávislosti na konkrétní poloze roztoče na podložce. Následně jsme si ukládali aktuální hodnotu accuracy, F1 score a loss, obdobně jsme měřili tyto hodnoty i ve validační části. Konečná úspěšnost klasifikace vyšla 98,38 %.

Po trénovací a validační fázi jsme použili na sešitou spadovou podložku blob detektor. Detekované objekty byly vloženy do natrénované sítě, která pro každý vstupní obrázek určila, zda se na obrázku nachází kleštík a s jakou pravděpodobností je tato predikce správná. Dle těchto závěrů jsme na vloženou podložku vykreslili kružnice kolem všech detekovaných objektů z blob detektoru. Barva kružnic odpovídala výsledkům klasifikace. Modře a fialově byli zvýrazněni klešťíci včelí, červeně a zeleně jiné objekty.

Výsledky trénování a následné validace sítě ukázaly, že naše síť dosahuje hodnot dostačujících pro detekci kleštíka včelího (vysoké precission a téměř zanedbatelného loss). MobilenetV3 tedy představuje vhodný model pro vytvoření aplikace pro monitoring varroázy ve včelstvu. Naopak detekce pomocí klasických metod zpracování obrazu nevykazuje dostatečnou přesnost.

Seznam použité literatury

[1] O. Boecking and E. Generschr, "Varroosis – the Ongoing Crisis in Bee Keeping," 2008.

[2] J. D. Ellis and C. Z. Nalen, "Varroa mite," 2010. [Online]. Available: http://entnemdept.ufl.edu/creatures/misc/bees/varroa_mite.htm

[3] www.vcelky.cz, "Hustota zavčelení České republiky," 2016. [Online]. Available: http://vcelky.cz/clanky/2016-hustota-zavceleni-ceske-republiky.htm

[4] Časopis MODERNÍ VČELAŘ, "Zpráva o stavu zemědělství ČR v roce 2019," 2020. [Online]. Available: https://www.modernivcelar.eu/8163-zprava-o-stavu-zemedelstvi-cr-v-roce-2019

[5] VETHO-PHARMA, "Varroa průvodce," 2019. [Online]. Available: https://www.mhvet.cz/data/files/e3f25bbd545480aaa28b12a69d9e8a10.pdf

[6] Varroáza.cz, "Varroáza - parazitní onemocnění včely medonosné." [Online]. Available: http://www.varroaza.cz

[7] Státní veterinární správa, "Metodika kontroly zdraví zvířat a nařízené vakcinace na rok 2019," 2019. [Online]. Available: https://www.svscr.cz/wp-content/files/dokumenty-a-publikace/Dokument-67435-2018-MZE-17212.pdf

[8] mojevcely.eu, "Biologie kleštíka," 2009. [Online]. Available: https://www.mojevcely.eu/clanky/varrooza/biologie-klestika/

[9] Výzkumný ústav včelařský, "Celý rok proti varroáze." [Online]. Available: https://www.beedol.cz/cely-rok/

[10] ——, "Organické kyseliny." [Online]. Available: https://www.beedol.cz/leceni/organicke-kyseliny/

[11] Varroáza.cz, "Biologie varroázy." [Online]. Available: http://www.varroaza.cz/biologie-varroazy/

[12] Výzkumný ústav včelařský, "Ochrana dlouhověkých včel." [Online]. Available: https://www.beedol.cz/leceni/ochrana-dlouhovekych-vcel/

[13] Med a včely, "Izolace včelích matek aneb jak pomoci klíckování bojovat s varroázou," 2019. [Online]. Available: http://www.vcely-med.cz/izolace-vcelich-matek

[14] Výzkumný ústav včelařský, "Monitoring varroázy." [Online]. Available: https://www.beedol.cz/monitoring/

[15] C. Harris and M. Stephens, in *[Procedings of the Alvey Vision Conference 1988.*

[16] Z. Wang and Z. Yang, "Review on image-stitching techniques," *Multimedia Systems*, vol. 26, no. 4, pp. 413–430, 2020. [Online]. Available: http://link.springer.com/10.1007/s00530-020-00651-y

[17] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. [Online]. Available: http://link.springer.com/10.1023/B:VISI.0000029664.99615.94

[18] D. Tyagi, "Introduction to SIFT (Scale Invariant Feature Transform)," 2019. [Online]. Available: https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

[19] i2 tutorials contributors, "What are SIFT and SURF?" 2019. [Online]. Available: https://www.i2tutorials.com/what-are-sift-and-surf/

[20] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF - Speeded Up Robust Feature," in *Springer Berlin Heidelberg*, 2006.

[21] D. Tyagi, "Introduction to SURF (Speeded-Up Robust Features)," 2019. [Online]. Available: https://medium.com/data-breach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e

[22] Wikipedia contributors, "Haar Wavelet," 2021. [Online]. Available: https://en.wikipedia.org/wiki/Haar_wavelet

[23] M. Brown and D. G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features," *International Journal of Computer Vision*, 2007.

[24] S. Mallick, "Blob Detection Using OpenCV (Python, C++)," 2015. [Online]. Available: https://learnopencv.com/blob-detection-using-opencv-python-c/

[25] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[26] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[27] scikit-image contributors, "Image processing in Python," 2014. [Online]. Available: https://scikit-image.org/docs/0.3/auto_examples/plot_hough_transform.html

[28] Q. Ji and Y. Xie, "A new efficient ellipse detection method," in *Proceedings of the International Conference on Pattern Recognition*, 2002, pp. 957–960.

[29] B. C. Durak, "Artificial Neural Networks," 2017. [Online]. Available: https://wiki.tum.de/display/lfdv/Artificial+Neural+Networks

[30] K. Connor, "Why are Neuron Axons Long and Spindly?" 2018. [Online]. Available: https://ucsdnews.ucsd.edu/pressrelease/why_are_neuron_axons_long_and_spindly

[31] M. Straka, “Hluboké učení,” 2021, přednáška, Matematicko-fyzikální fakulta, Univerzita Karlova. [Online]. Available: https://ufal.mff.cuni.cz/courses/npfl114/2021-summer

[32] A. Jareš and P. Martišek, “Šlechtíme roboty: Line follower pomocí evoluce,” 2018. [Online]. Available: http://robodoupe.cz/2018/slechtime-roboty-line-follower-pomoci-evoluce/

[33] A. Biswal, “An Introduction To Deep Learning With Python,” 2021. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-with-python?source=sl_frs_nav_user_clicks_on_next_tutorial

[34] V. Bushaev, “How do we ‘train’ neural networks?” 2017. [Online]. Available: https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73

[35] M. Nielsen, “Neural Networks and Deep Learning,” 2019. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap1.html#the_architecture_of_neural_networks

[36] S. Balaji, “Binary Image classifier CNN using TensorFlow,” 2020. [Online]. Available: https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697

[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[38] A. Hossain and S. A. Sajib, “Classification of Image using Convolutional Neural Network (CNN),” 2019.

[39] F. Culfazi, “Transfer Learning using Mobilenet and Keras,” 2018. [Online]. Available: https://towardsdatascience.com/transfer-learning-using-mobilenet-and-keras-c75daf7ff299

[40] A. Howard and M. Sandler, “Searching for MobileNetV3,” 2019.

Seznam obrázků