



# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

## ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

# POČÍTÁNÍ CHARAKTERISTICKÝCH ŠUPIN JEŠTĚRKY OBECNÉ V BAREVNÝCH OBRAZECH

COUNTING OF CHARACTERISTIC SCALES OF SAND LIZARDS IN COLOUR IMAGES

### DIPLOMOVÁ PRÁCE

MASTER'S THESIS

### AUTOR PRÁCE

AUTHOR

**Bc. Štěpán Maršala**

### VEDOUCÍ PRÁCE

SUPERVISOR

**Ing. Pavel Škrabánek, Ph.D.**

**BRNO 2022**



## Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	<b>Bc. Štěpán Maršala</b>
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	<b>Ing. Pavel Škrabánek, Ph.D.</b>
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

### **Počítání charakteristických šupin ještěrky obecné v barevných obrazech**

#### **Stručná charakteristika problematiky úkolu:**

Počítání objektů v obraze je často řešeným úkolem v řadě technických i netechnických odvětví. Počítání charakteristických šupin ještěrky obecné za účelem určení jejího pohlaví se však od běžně řešených úkolů odlišuje v několika významných bodech: a) charakteristické šupiny jsou určeny tvarem i polohou vůči tělu ještěrky, b) charakteristické šupiny mají velmi podobný tvar jako ostatní šupiny, c) charakteristické šupiny mají různou velikost, d) počítání charakteristických šupin je nutné realizovat zvlášť pro levou a zvlášť pro pravou stranu.

#### **Cíle diplomové práce:**

Student vypracuje rešerši mapující techniky využívané k detekci objektů a segmentaci obrazu. Student analyzuje problém, navrhne a implementuje systém pro počítání charakteristických šupin v barevných obrazech, anotuje zadanou datovou sadu, a vyhodnotí efektivitu vytvořeného řešení.

#### **Seznam doporučené literatury:**

EPLANOVA, Galina V. a Evgeny S. ROITBERG. Sex identification of juvenile sand lizards, *Lacerta agilis* using digital images. *Amphibia-Reptilia* [online]. 2015, 36(3), 215-222 [cit. 2021-9-6]. ISSN 0173-5373. Dostupné z: doi:10.1163/15685381-00002996.

GONZALEZ, Rafael C. a Richard E. WOODS. *Digital image processing*. New York, NY: Pearson, [2018]. ISBN 978-0133356724.

ZHAO, Zhong-Qiu, Peng ZHENG, Shou-Tao XU a Xindong WU. Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems* [online]. 2019, 30(11), 3212-3232 [cit. 2020-10-08]. ISSN 2162-237X. Dostupné z: doi:10.1109/TNNLS.2018.2876865.

LATEEF, Fahad a Yassine RUICHEK. Survey on semantic segmentation using deep learning techniques. Neurocomputing [online]. 2019, 338, 321-348 [cit. 2019-09-03]. DOI: 10.1016/j.neucom.2019.02.003. ISSN 09252312. Dostupné z: <https://linkinghub.elsevier.com/retrieve/pii/S092523121930181X>.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

---

doc. Ing. Radomil Matoušek, Ph.D.  
ředitel ústavu

---

doc. Ing. Jaroslav Katolický, Ph.D.  
děkan fakulty

## **ABSTRAKT**

V diplomové práci je popsán návrh a implementace programu pro počítání sekundárních šupin v obrazových datech ventrálních stran těl ještěrek obecných. Program respektuje požadavky vědců z Ústavu biologie obratlovců Akademie věd ČR a Pedagogické fakulty Masarykovy univerzity na ovladatelnost a na přesnost výsledků. Program se skládá z několika částí. Na vstupu přijímá fotografie ještěrek obecných, ve kterých vyřízne zájmovou oblast. Orientaci těchto výřezů unifikuje pomocí detekovaných objektů. Detekci objektů zajišťuje YOLOv4. Další část programu zvaná Centroid detektor určuje v unifikovaných výřezech pozice středů sekundárních šupin. Tato část využívá konvoluční neuronové sítě U-Net, která je speciálně modifikovaná pro detekci středů objektů v těsné blízkosti. Poslední části programu rozdělí detekované pozice středů šupin na levou a pravou sekundární řadu a zapíše jejich počty do výstupního souboru.

## **ABSTRACT**

The diploma thesis describes the design and implementation of a program for counting secondary scales in the image data of the ventral sides of the bodies of sand lizards. The program respects the requirements of scientists from the Institute of Vertebrate Biology of the Czech Academy of Sciences and the Faculty of Education at Masaryk University for the controllability and accuracy of results. The program consists of several parts. In input receives photos of sand lizards, in which he cuts out an area of interest. Unifies the orientation of these sections using detected objects. Object detection is provided by YOLOv4. Another part of the program called the Centroid Detector determines the position of the centers of the secondary scales in the unified sections. This part uses the U-Net convolutional neural network, which is specially modified to detect the centers of objects in close proximity. The other parts of the program divide the detected positions of the scale centers into left and right secondary rows and write their numbers to the output file.

## **KLÍČOVÁ SLOVA**

strojové vidění, konvoluční neuronové sítě, YOLOv4, Centroid detektor, Python 3

## **KEYWORDS**

machine vision, convolutional neural networks, YOLOv4, Centroid Detector, Python 3





2022

## BIBLIOGRAFICKÁ CITACE

MARŠALA, Štěpán. *Počítání charakteristických šupin ještěrky obecné v barevných obrazech*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, 2022, 101 s. Diplomová práce. Vedoucí práce: Ing. Pavel Škrabánek, Ph.D.





## ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato diplomová práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků.

V Brně dne 20. 5. 2022

.....

Štěpán Maršala



## PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Pavlu Škrabánkovi, Ph.D. za velmi přátelský a pohotový přístup. Dále bych chtěl poděkovat doc. Mgr. Natálii Martínkové, Ph.D. a Ing. Radovanu Smolinskému, Ph.D. za poskytnutí datové sady a za odborné konzultace.



# OBSAH

<b>1</b>	<b>ÚVOD</b> .....	<b>15</b>
<b>2</b>	<b>VYMEZENÍ CÍLE PRÁCE</b> .....	<b>17</b>
2.1	Požadavky .....	18
2.2	Návrh řešení .....	18
<b>3</b>	<b>REŠERŠE POUŽITÝCH METOD</b> .....	<b>21</b>
3.1	Obecné přístupy k počítání podobných objektů v obraze .....	21
3.2	Umělé neuronové sítě .....	21
3.2.1	Obecný popis umělých neuronových sítí .....	22
3.2.2	Učení .....	24
3.2.3	Techniky trénování .....	25
3.3	Digitální obrazová data .....	26
3.4	Konvoluční neuronové sítě .....	26
3.4.1	Diskrétní dvourozměrná konvoluce .....	27
3.4.2	Popis konvolučních neuronových sítí .....	27
3.4.3	Augmentace .....	30
3.5	Klasifikace obrazu .....	34
3.6	Detekce objektů .....	35
3.6.1	mean Average Precision .....	36
3.6.2	YOLO .....	36
3.7	Segmentace obrazu .....	38
3.7.1	U-Net .....	39
3.8	Centroid detektor .....	41
3.8.1	Generátor lokalizačních map .....	41
3.8.2	Centroid counterpoint .....	42
3.9	Souborové učení .....	44
<b>4</b>	<b>ŘEŠENÍ</b> .....	<b>45</b>
4.1	Popis dostupné datové sady a její analýza .....	45
4.2	Popis navrženého řešení .....	46
4.2.1	Detekce hlavy, břicha a ocasu .....	48
4.2.2	Unifikace orientace a vyřiznutí zájmové oblasti .....	49
4.2.3	Detekce sekundárních šupin .....	50
4.2.4	Rozdělení detekovaných šupin na pravé a levé .....	55
4.2.5	Spočítání pravých a levých šupin .....	56
4.3	Příprava datových sad .....	57
4.3.1	Štítkování hlavy, břicha a ocasu .....	57
4.3.2	Štítkování šupin .....	58
4.4	Implementace řešení .....	60

4.4.1	Trénování modelu YOLOv4 .....	60
4.4.2	Trénování modelů U-Net .....	62
<b>5</b>	<b>VÝSLEDKY A JEJICH VYHODNOCENÍ .....</b>	<b>67</b>
5.1	Výběr modelů U-Net .....	67
5.2	Výsledný program .....	69
5.2.1	Lokální verze pro Windows .....	69
5.2.2	Cloudová verze .....	70
5.3	Vyhodnocení .....	70
5.3.1	Přesnost řešení .....	72
5.3.2	Rychlost výpočtu .....	72
<b>6</b>	<b>DISKUSE .....</b>	<b>75</b>
<b>7</b>	<b>ZÁVĚR .....</b>	<b>77</b>
	<b>SEZNAM POUŽITÉ LITERATURY .....</b>	<b>79</b>
	<b>SEZNAM ZKRATEK .....</b>	<b>85</b>
	<b>SEZNAM SYMBOLŮ .....</b>	<b>87</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>91</b>
	<b>SEZNAM TABULEK .....</b>	<b>95</b>
	<b>SEZNAM PŘÍLOH .....</b>	<b>97</b>

# 1 ÚVOD

Ještěrka obecná (*Lacerta agilis Linnaeus*, 1758) žije v různých typech prostředí mírného podnebného pásu euroasijského kontinentu. Je důležitou součástí potravního řetězce. Patří k plazům, u kterých se pohlaví vylíhnutých jedinců určuje teplotou prostředí. Poměr pohlaví nedospělých ještěrek nemusí odpovídat poměru pohlaví starších jedinců. Příčinou je odlišné chování samců a samic, které vede k různé zranitelnosti před predátory. Asi jen 2 % nově narozených jedinců žijí déle než jeden rok [1].

Pro sledování klimatických změn a změn poměrů v potravinovém řetězci vědci z Ústavu biologie obratlovců Akademie věd ČR (ÚBO AV ČR v.v.i.) a Pedagogické fakulty Masarykovy univerzity (PdF MUNI) dlouhodobě monitorují poměr pohlaví juvenilních a subadultních jedinců (dále jen nedospělých ještěrek). Problémem je, že u nedospělých ještěrek nelze pohlaví určit jednoznačně podle sekundárních pohlavních znaků (zbarvení, velikost hlavy, přítomnosti femorálních pórů atd.). Pohlaví nedospělých ještěrek vědci v současnosti zjišťují pomocí analýzy DNA [2], která je nákladná a časově velmi náročná.

Výzkum Eplanové a Roitberga ukázal korelaci mezi počtem sekundárních šupin na ventrální straně těla (na bříše) jedince a jeho pohlavím [3]. Vědci z ÚBO AV ČR v.v.i. a PdF MUNI ještěrky dlouhodobě fotografují a vedou záznamy o pohlaví dospělých jedinců. Dostupnost těchto záznamů umožňuje vytvořit automatizovaný systém, který dokáže predikovat pohlaví všech vyfocených jedinců, tedy i těch nedospělých (pokud jsou na fotografii viditelné řady sekundárních šupin). Základním předpokladem takového systému je schopnost přesně spočítat sekundární šupiny v barevných fotografiích ventrálních stran ještěrek obecných.

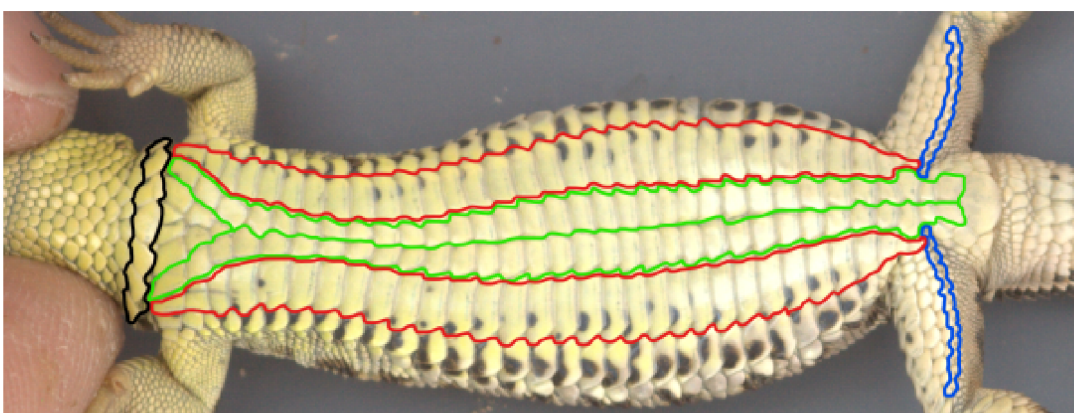
Cílem diplomové práce je navrhnout a implementovat funkční program pro počítání sekundárních šupin v obrazových datech ventrálních stran těl ještěrek obecných. Práce neřeší určení pohlaví jedinců na fotografiích ani počet jiných než sekundárních šupin.



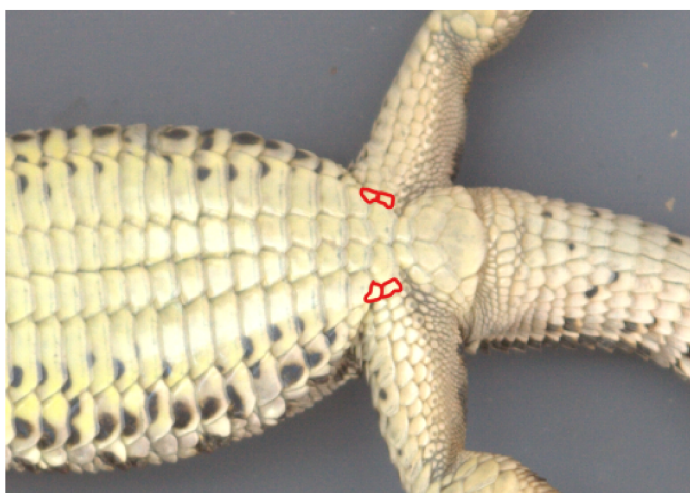


## 2 VYMEZENÍ CÍLE PRÁCE

Primární šupiny se nacházejí u linie středu těla na ventrální straně těla ještěrky (vyznačeno zeleně na obr. 1). Sekundární šupiny sousedí s primárními a jsou ve dvou pásech na levé a pravé straně (vyznačeno červeně na obr. 1). Počítají se od límce (vyznačeno černě na obr. 1) až po šupiny s femorálními póry na zadních končetinách (vyznačeno modře na obr. 1). Počet sekundárních šupin na levé a pravé straně nemusí být stejný. Ventrální strana ještěrky je často skvrnitá. Variabilitu počtu sekundárních šupin výrazně ovlivňuje posledních několik šupin před šupinami s femorálními póry, které jsou vyznačeny červeně na obr. 2. Ty jsou velmi malé a obtížně rozpoznatelné.



Obr. 1: Vyznačení primárních šupin (zeleně), sekundárních šupin (červeně), límce (černě) a šupin s femorálními póry (modře) na ventrální straně těla ještěrky obecné. Převzato a upraveno z [4].



Obr. 2: Detail (červeně) posledních ventrálních sekundárních šupin. Převzato a upraveno z [4].

## 2.1 Požadavky

Po konzultaci s vědci z ÚBO AV ČR v.v.i. a PdF MUNI byly na výsledné řešení v podobě ovladatelného programu vysloveny následující požadavky:

- Program musí být ovladatelný technicky mírně pokročilým uživatelem, který umí pracovat s příkazovým řádkem.
- Vstupem do programu jsou neupravené fotografie z fotoaparátu nebo z databáze ve formátech jpg, psd a CR2.
- Program výsledky uloží do txt souboru, kde bude každý řádek odpovídat jedné fotografii. Bude obsahovat informace v následujícím pořadí oddělené tabulátorem:
  - cesta k fotografii,
  - počet detekovaných sekundárních šupin na levé straně,
  - počet detekovaných sekundárních šupin na pravé straně
  - a informace o zpracování (validita apod.).
- Výsledky budou vizualizovány v podobě nových obrazových dat ještěrek s vyznačenými pozicemi detekovaných šupin. Tato funkce je požadována zejména kvůli kontrole a analýze výsledků.
- Program musí detekovat i poslední ventrální sekundární šupiny.
- Program na základě detekovaného počtu šupin posoudí validitu výsledků pro každou fotografii.
- Roitberg a Eplanova [3] uvádějí, že rozdíl počtu sekundárních šupin u samců a samic je 2–3 na jednu sekundární řadu. To znamená, že 4–6 šupin na obě sekundární řady. Po konzultaci s vědci z ÚBO AV ČR v.v.i. a PdF MUNI bylo dohodnuto, že průměrná odchylka skutečného počtu sekundárních šupin od detekovaného může být maximálně 2 šupiny na obě řady. Naměřené odchylky budou popsány relevantními statistickými metodami pro zhodnocení výsledků a přesnosti řešení.

Program, který bude funkční a zároveň bude vyhovovat všem uvedeným požadavkům, bude validním výsledným řešením. Návrh, implementace a zhodnocení výsledného řešení povede k úspěšnému naplnění cíle této diplomové práce.

## 2.2 Návrh řešení

Program bude obsahovat systém strojového vidění, který na vstupu přijme fotografii ještěrky a na výstupu vrátí počet sekundárních šupin. Základem systému bude Centroid detektor, který z obrazových dat ještěrky vytvoří souřadnice hledaných šupin. Vstupní fotografie však bude potřeba co nejvíce homogenizovat tak, aby byl Centroid detektor co nejpřesnější. Homogenizací fotografií se myslí například jednotné

rozměry, oříznutí nepotřebných částí a orientace jedním směrem. Ke správné homogenizaci bude potřeba určit polohu jednotlivých částí ještěrky vhodným detektorem objektů. Určení počtu a polohy šupin může být docíleno například vhodnou obrazovou transformací. Při návrhu Centroid detektoru a detektoru objektů se využívá hluboké učení. Aby byly detektory robustnější, může být při trénování modelů použita augmentace vstupních obrazových dat či více natrénovaných modelů (souborové učení).

Protože je strojové učení relativně novým a rychle se měnícím vědním oborem, některé pojmy nejsou v odborné literatuře definované česky. V těchto případech budou v textu použité anglické výrazy, aby nedocházelo k nedorozumění.



## 3 REŠERŠE POUŽITÝCH METOD

Kapitola obsahuje teoretickou rešerši všech použitých metod, které jsou potřebné k návrhu systému strojového vidění.

### 3.1 Obecné přístupy k počítání podobných objektů v obraze

V obrazových datech bývá někdy potřeba spočítat určité podobné objekty nebo oblasti. Pokud nemají hledané objekty homogenní tvar, velikost nebo barvu, je často jedinou možností, jak je lokalizovat a spočítat, přistoupit k metodám hlubokého učení.

Obvyklým přístupem k tomuto problému je použití detektoru objektů. Detektory objektů nejsou účinné pro počítání podobných objektů, které jsou v obraze blízko u sebe (buňky, smítka, stromy z letadla apod.), vynikají však v detekci nehomogenních objektů různých tříd (pes, kočka, auto apod.) [5]. Hsieh, Lin a Hsu přišli s myšlenkou vylepšení detektorů objektů pro počítání homogenních objektů, které jsou umístěné v řadách [6]. U objektů shlukujících se v pravidelných tvarech se dá prioritizovat prostor pro hledání nových objektů. Algoritmus dokonce implementovali pro počítání aut na parkovištích v reálném čase. Dokázali tak důležitost vhodného výběru prohledávaných oblastí.

Počítání objektů se dá realizovat i segmentací obrazu. Mezei a Darabant aplikovali segmentaci pro počítání chodců [7]. Ačkoli jejich algoritmus správně spočítal 92 % lidí, v závěru práce uvádějí, že velmi špatně zvládal skupiny chodců jdoucích velmi blízko u sebe. Nizozemská studie, která se zabývala počítáním stromů z leteckých snímků, prokázala účinnost kombinace segmentace obrazu a následné vykreslení vektorového pole, které ukazuje na středy hledaných lokací (CentroidNet), oproti detektoru YOLOv3. [8].

Pro počítání objektů v obraze se nedá určit jeden účinný způsob. Ačkoli se může zdát, že známé detektory objektů, jako je například YOLO, jsou pro tyto typy úloh téměř univerzální, pro některé aplikace bývají účinnější specificky navržené detektory.

### 3.2 Umělé neuronové sítě

Tato práce využívá umělých neuronových sítí, což je podoblast strojového učení. Strojové učení je podoblastí umělé inteligence. Jde o algoritmy, které mohou zpřesňovat své výsledky bez zásahu člověka za použití dat [9].

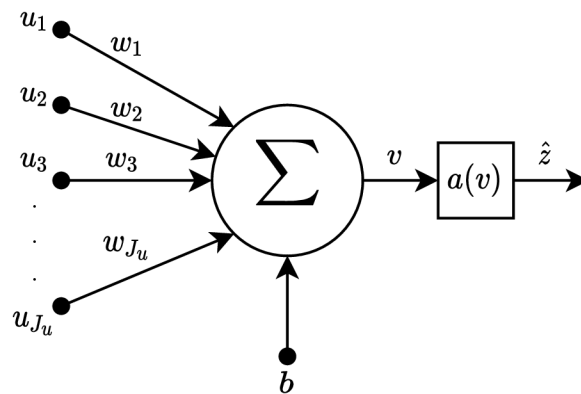
### 3.2.1 Obecný popis umělých neuronových sítí

Umělá neuronová síť je metoda inspirovaná strukturou neuronů v biologickém mozku. Poprvé ji popsal McCulloch v roce 1943 [10]. Slouží především k řešení složitých výpočetních operací, které je náročné provádět analyticky. Neuronová síť se skládá z na sebe napojených vrstev neuronů. Koncept umělého neuronu poprvé definoval Frank Rosenblatt jako perceptron v roce 1958 [11].

V současnosti používaný umělý neuron, vizualizovaný na obr. 3, přijímá na vstupu několik hodnot  $u$  a vrací výstupní hodnotu

$$\hat{z} = a \left( \sum_{i=1}^{J_u} (w_i \cdot u_i) + b \right), \quad (1)$$

kde  $a(\cdot)$  je aktivační funkce,  $J_u$  je počet vstupů,  $w$  jsou váhy jednotlivých propojení,  $b$  je bias neboli prahová hodnota aktivace neuronu. Hodnota  $v$  na obr. 3 představuje sumu všech součinů vah  $w_i$  s hodnotami na ně napojených vstupů  $u_i$  a biasu  $b$ .



Obr. 3: Model umělého neuronu odpovídající rovnici (1).

Aktivační funkce  $a(\cdot)$  transformuje hodnotu  $v$  na výstupní signál  $\hat{z}$ . Mezi používané aktivační funkce zobrazené na obr. 4 patří funkce sigmoid

$$a(v) = \frac{1}{1 + e^{-v}}, \quad (2)$$

funkce RELU

$$a(v) = \max(0, v), \quad (3)$$

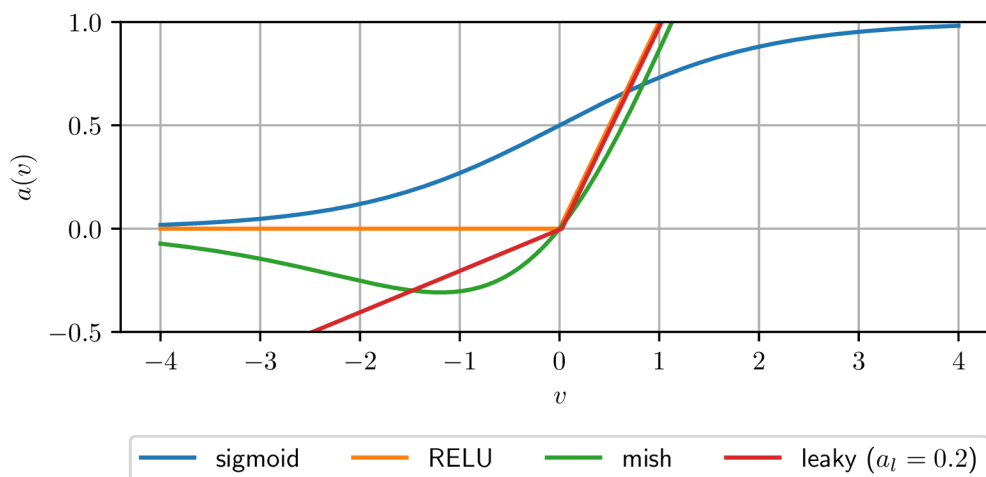
funkce mish [12]

$$a(v) = v \cdot \tanh(\ln(1 + e^v)), \quad (4)$$

a funkce leaky

$$a(v) = v, \text{ pro } v \in (-\infty, 0); f(v) = a_l \cdot v, \text{ pro } v \in \langle 0, \infty \rangle, \quad (5)$$

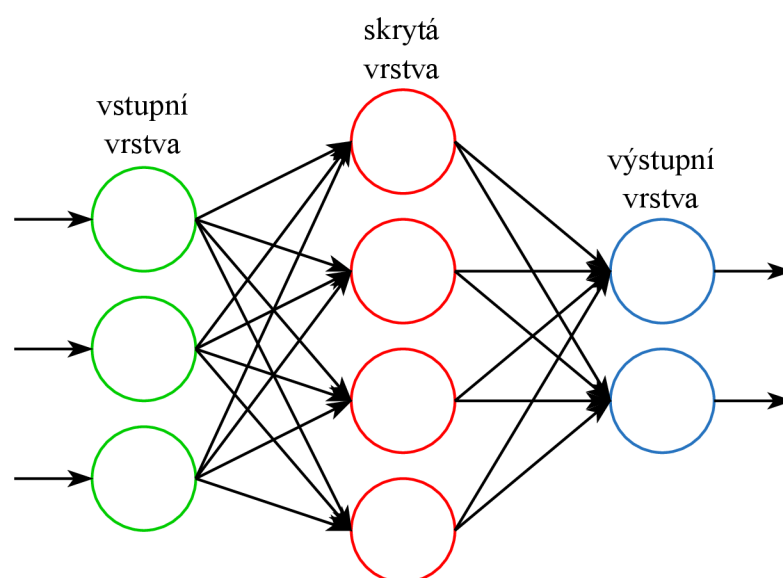
kde  $a_l$  je volitelný parametr záporné části.



Obr. 4: Různé typy aktivačních funkcí.

Umělá neuronová síť spojuje více neuronů uspořádaných ve vrstvách. První vrstva je vstupní a přijímá vstupní hodnoty  $u$ . Další vrstva přijímá výstupy  $\hat{z}$  neuronů vstupní vrstvy a následující vrstvy vždy přijímají výstupy neuronů vrstev předchozích. Tyto vrstvy se nazývají skryté. Skrytých vrstev může být v síti více, stejně tak jako neuronů ve všech vrstvách. Každý neuron přijímá výstupy ze všech neuronů předchozí vrstvy. Poslední vrstva se nazývá výstupní. Hodnoty  $\hat{z}$  neuronů výstupní vrstvy jsou výstupy celé umělé neuronové sítě. Příklad neuronové sítě je na obr. 5.

Uspořádání vrstev, propojení vrstev, počty a vlastnosti neuronů pak definují architekturu sítě. Pokud má umělá neuronová síť desítky vrstev, hovoříme o hluboké neuronové síti. Hluboké neuronové sítě se používají k řešení složitých problémů. Nevýhodou je, že nelze analyzovat jejich vnitřní způsob výpočtu.



Obr. 5: Příklad umělé neuronové sítě s jednou skrytou vrstvou.

### 3.2.2 Učení

Schopnost umělé neuronové sítě řešit určitý typ problému nezávisí jen na architektuře, ale i na hodnotách vah  $w$  mezi neurony. Hodnoty vah je potřeba adaptovat učním neboli trénováním. Učení probíhá za pomoci datové sady (datasetu), což je množina vzorků (samples). Vzorky jsou příklady hodnot vstupů, které mohou doprovázet i očekávané hodnoty výstupů, které by měla umělá neuronová síť vracet na výstupu. Datovou sadu dělíme na trénovací sadu, která je určená k učení, a testovací sadu, pomocí které se po ukončení učení vyhodnocuje, jak přesně odpovídají výstupy umělé neuronové sítě očekávaným hodnotám.

Dva základní typy učení jsou učení s učitelem a učení bez učitele. Učení bez učitele znamená učit neuronovou síť bez zpětné vazby, tedy bez očekávaných hodnot výstupů. Příkladem takového učení je shluková analýza. Umělé neuronové síti se předkládají polohy bodů v prostoru, které jsou v závislosti na architektuře rozděleny do shluků. Pokud síť učíme s učitelem, znamená to, že poskytujeme modelu síti zpětnou vazbu, tedy vzorky včetně očekávaného výstupu.

Matematici se dlouho zabývali vhodným způsobem učení neuronových sítí. V roce 1974 popsal P. Werbos ve své disertační práci první použitelný algoritmus pro trénování umělých neuronových sítí nazývaný zpětné šíření chyby (backpropagation) [13] a v roce 1982 jej zdokonalil [14]. Princip tohoto algoritmu se používá do dnes. Algoritmus zpětného šíření chyby je rekurzivní algoritmus, který srovná hodnoty aktivačních funkcí neuronů výstupní vrstvy po reakci sítě na určitý vzorek ve vstupní vrstvě s očekávaným výstupem (učení s učitelem). Pomocí ztrátové funkce  $C(\cdot)$  (loss function) je vypočítána chyba sítě. Existují různé ztrátové funkce, každá se používá na jiný typ problému. Nejznámější je kvadratická ztrátová funkce

$$C = (\hat{z} - z)^2, \quad (6)$$

kde  $z$  je očekávaná hodnota výstupu. Hodnota ztrátové funkce je vypočítána pro všechny neurony výstupní vrstvy a znamená, jak by se měla změnit hodnota každého neuronu, aby byl výstup sítě správný. Hodnoty aktivačních funkcí neuronů jsou však závislé na vahách spojení s neurony v předchozí vrstvě. Rovnice algoritmu zpětného šíření chyby určuje novou hodnotu váhy

$$w_i = w_{i-1} - \mu \frac{\partial C}{\partial w_{i-1}}, \quad (7)$$

kde  $w_i$  je nová hodnota váhy,  $w_{i-1}$  je předcházející hodnota váhy a  $\mu$  je learning rate.

V algoritmu zpětného šíření chyby jde o minimalizaci ztrátové funkce. Čím menší je  $\mu$ , tím menší jsou změny vah ve směru lokálního minima ztrátové funkce. Způsobů, jakými algoritmus zpětného šíření chyby upravuje váhy pomocí parametru  $\mu$ , je více a označujeme je jako optimalizační algoritmy (optimizers). Výše pospaný



optimalizační algoritmus (7) se nazývá gradientní sestup (gradient descent). Pokud použijeme jen část vzorků, které jsou určeny k trénování, hovoříme o stochastickém gradientním sestupu (Stochastic Gradient Descent, SGD), který je výpočetně méně náročný. Optimalizační algoritmy mohou být doplněny o momentovou větu, neboli hybnost (momentum). Hybnost závisí na tom, jak moc se váha  $w$  mění (podobně jako ve fyzice). Stochastický gradientní sestup s momentovou větou je dán jako

$$w_i = w_{i-1} - \mu \nabla C(w_{i-1}) + \alpha \cdot w_{i-1}, \quad (8)$$

kde  $\alpha$  je konstanta momentové věty, pro kterou platí  $\alpha \in (0, 1)$  [15]. Častěji používaný optimalizační algoritmus publikovaný v roce 2014 Kingmou a Baem je Adam [16]. Adam upravuje váhy nejen pomocí parametru  $\mu$ , ale i podle hybnosti podobně jako SGD s momentovou větou. Jeho zjednodušený zápis je dán jako

$$w_i = w_{i-1} - \mu \frac{\hat{m}_{w_{i-1}}}{\sqrt{\hat{v}_{w_{i-1}} + \epsilon}}, \quad (9)$$

kde  $\hat{m}_w$  a  $\hat{v}_w$  jsou funkce hybnosti a derivace ztrátové funkce podle váhy a  $\epsilon$  je velmi malý parametr, který zajišťuje, aby nedošlo k dělení nulou [16].

Nové váhy se principem zpětného šíření chyby postupně vypočítají mezi všemi neurony umělé neuronové sítě. Algoritmus zpětného šíření chyby více upravuje hodnoty vah spojení mezi neurony s vysokou hodnotou ztrátové funkce a neurony, jejichž hodnota aktivační funkce výrazně ovlivňuje daný neuron. Tomuto principu se říká Hebbovské učení, které poprvé popsal D. O. Hebb v roce 1949 [17].

### 3.2.3 Techniky trénování

Pokud spočítáme změny vah celé umělé neuronové sítě pro soubor několika vzorků najednou, a až poté upravíme váhy, nazveme soubor vzorků dávkou (batch). Trénování pomocí dávek zrychluje učení díky nižší výpočetní náročnosti. Jakmile optimalizační algoritmus použije k trénování celou trénovací sadu, tedy soubor všech vzorků, vykoná jednu epochu. V každé epoše je trénovací sada rozdělená na trénovací a evaluační podmnožinu. Rozdělení proběhne předem (může probíhat i před každou epochou) a trénuje se jen na trénovací podmnožině.

Po každé epoše se vypočítá hodnota metriky vzorků z evaluační podmnožiny. Metrika může být například hodnota ztrátové funkce. Metrika slouží pouze pro posuzování natrénovaných vah, nikoliv pro jejich úpravu. Hodnota metriky může být použita pro ukončení trénování. Pokud začne stoupat, hovoříme o nežádoucím přetrénování sítě a trénování se zastaví. Mezi další metody ukončení trénování patří například maximální počet epoch.

Vedle trénovací sady existuje sada testovací, která nemá vliv na trénování, ani na jeho ukončení. Výpočtem metriky testovací sady po ukončení trénování modelu hodnotíme úspěšnost trénování.

### 3.3 Digitální obrazová data

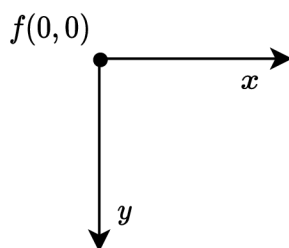
Digitální obraz vzniká osvětlením světlocitného čipu, který je tvořen mřížkou fotodiód o rozlišení  $M \times N$  [18]. Tím se obraz z reálného světa navzorkuje. Každá fotodioda se během osvětlení nabije nábojem o určité hodnotě. Hodnota náboje je převedena na napětí, které je pomocí A/D převodníku kvantováno na digitální hodnotu. Rozmezí digitálních hodnot je určeno počtem bitů na jeden kvantovaný element, který se nazývá pixel. Nejčastější počet bitů je 8, pixel tak může nabývat hodnot 0 až  $2^8 - 1$ . Hodnoty pixelů jsou uspořádány do matice o rozměrech  $M \times N$ , což odpovídá uspořádání fotodiód. Tato matice se popisuje obrazovou funkcí

$$f(x, y) = \begin{bmatrix} f(0, 0) & \cdots & f(0, N - 1) \\ \vdots & \ddots & \vdots \\ f(M - 1, 0) & \cdots & f(M - 1, N - 1) \end{bmatrix}, \quad (10)$$

kde  $f(\cdot)$  je obrazová funkce a  $(x, y)$  jsou prostorové souřadnice obrazové funkce  $f(\cdot)$  [18].

Obrazová data reprezentována v každém pixelu obrazové funkce jednou hodnotou jsou však černobílá. Obrazová funkce  $f(\cdot)$  barevného obrazu se rozšiřuje v každém pixelu o hodnoty barevného prostoru. Nejčastěji používaným barevným prostorem je RGB (Red Green Blue), který má v každém bodě obrazové matice hodnoty pro červenou, zelenou a modrou barvu v rozmezí 0 až 255 (8 bitů).

V běžně používaných softwarech pro zpracování obrazu, jako je například volně dostupný OpenCV, se používá opačně orientovaný souřadný systém znázorněný na obr. 6, než jak jej definoval Gonzalez rovnicí (10).



Obr. 6: Ukázka souřadného systému používaného v softwaru OpenCV pro zpracování obrazu.

### 3.4 Konvoluční neuronové sítě

Konvoluční neuronové sítě se používají především ke zpracování obrazových dat. Principy konvolučního chování biologických neuronů poprvé popsali Hubel a Wiesel v roce 1959 na základě pozorování vizuálního vnímání koček [19]. Jejich práci se

inspiroval Kunihiko Fukushima a v roce 1980 popsal první konvoluční neuronovou síť [20]. Princip fungování je podobný umělým neuronovým sítím. Vrstvy neuronů jsou uspořádány do 3D numerických polí. Vstupní vrstva běžně bývá obrazová funkce. Konvoluční neuronové sítě ve skrytých vrstvách obsahují konvoluční vrstvy, vrstvy sjednocení, vrstvy s úplným propojením a mohou obsahovat i vrstvy rozšíření.

### 3.4.1 Diskrétní dvourozměrná konvoluce

V konvolučních vrstvách se využívá metoda zvaná diskretní dvourozměrná konvoluce. Jde o matematickou operaci, která vytváří ze vstupní obrazové funkce  $f(\cdot)$  o rozměrech  $M \times N$  a konvoluční masky  $h(\cdot)$  o rozměrech  $M_h \times M_h$  výstupní obrazovou funkci (mapu)

$$g(x, y) = f(x, y) * h(x, y) = \sum_{i=-M_h/2}^{M_h/2} \sum_{j=-N_h/2}^{N_h/2} f(x-i, y-j) h(i, j), \quad (11)$$

kde  $g(\cdot)$  je funkce popisující výstupní obrazovou funkci (mapu) a  $*$  je operátor diskretní konvoluce [18].

Rozměry výstupní obrazové funkce  $g(\cdot)$  se diskretní dvourozměrnou konvolucí zmenší z  $M \times N$  na rozměry  $M_g \times N_g$  dané rovnicemi

$$M_g = M - \frac{M_h + 1}{2}, \quad (12)$$

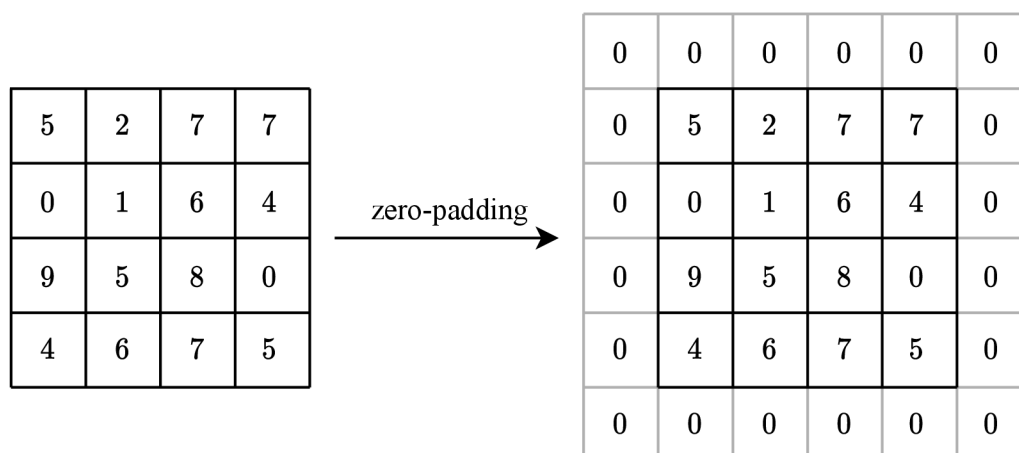
$$N_g = N - \frac{N_h + 1}{2}. \quad (13)$$

Aby zůstaly rozměry vstupní a výstupní obrazové funkce stejné, vstupní obrazová funkce  $f(\cdot)$  se po okrajích rozšíří o nuly. Tato operace znázorněná na obr. 7 se nazývá zero-padding. Rozšíření okrajů závisí na rozměrech konvoluční masky  $M_h \times M_h$ . Horizontální okraje se rozšíří o  $(M_h - 1)/2$  a vertikální o  $(N_h - 1)/2$ . Pokud mají být rozměry  $M_g \times N_g$  stejné jako  $M \times N$ , tak musí být hodnoty  $M_h$  a  $N_h$  vždy liché.

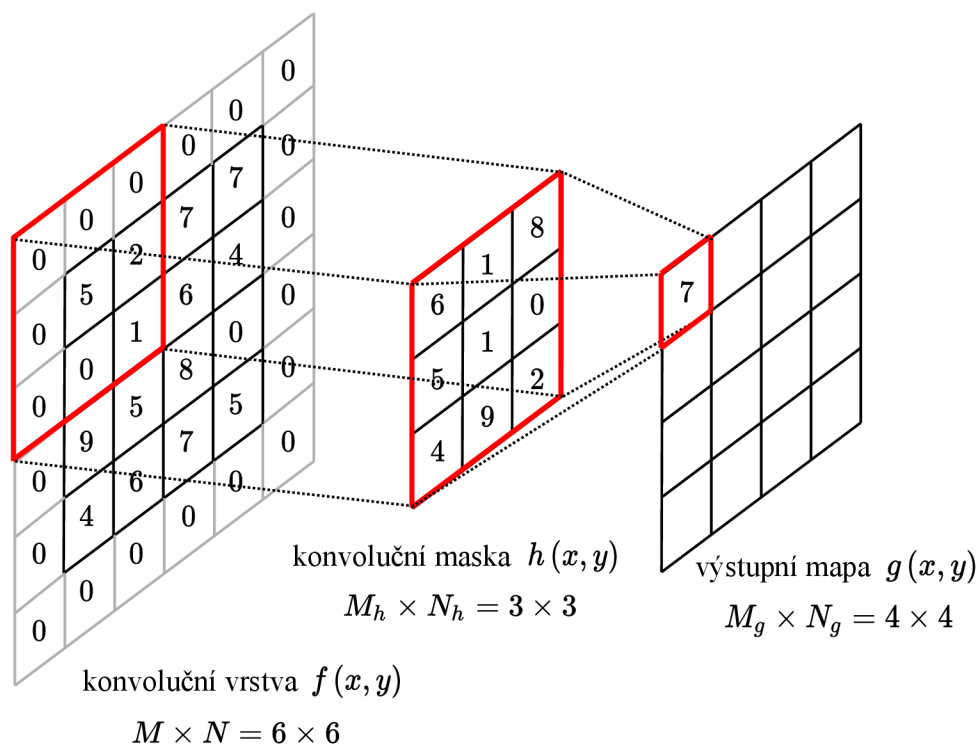
### 3.4.2 Popis konvolučních neuronových sítí

Konvoluční vrstvy jsou dvourozměrnou diskretní konvolucí transformovány na výstupní mapy  $g(\cdot)$ , tak jako je to ukázáno na obr. 8. Na konvoluční vrstvu se aplikuje konvoluční maska  $h(\cdot)$ . Pixely obrazových funkcí (konvolučních vrstev a výstupních map) jsou ekvivalentní neuronům v umělých neuronových sítích. Hodnoty v konvoluční masce by se pak daly chápat jako váhy spojení mezi neurony [18]. Rovnici pro diskretní dvourozměrnou konvoluci (11) doplníme o aktivační funkci  $a(\cdot)$ , prahovou hodnotu bias  $b$  a dostaneme vztah mezi neurony konvoluční vrstvy a výstupní mapy

$$g(x, y) = f(x, y) * h(x, y) = a \left( \sum_{i=-M_h/2}^{M_h/2} \sum_{j=-N_h/2}^{N_h/2} f(x-i, y-j) h(i, j) + b \right). \quad (14)$$

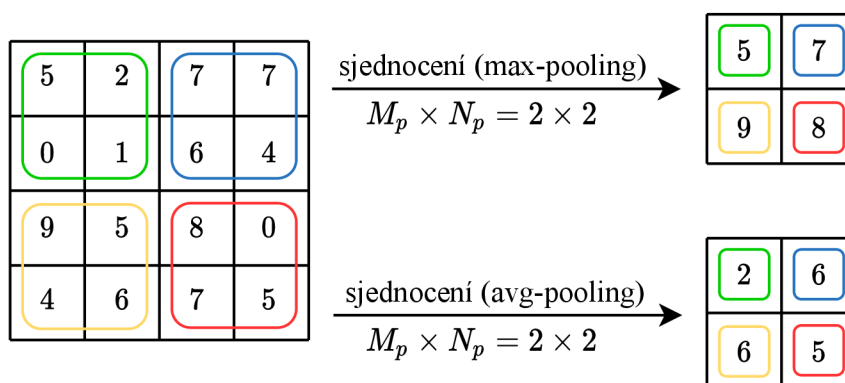


Obr. 7: Zero-padding obrazové funkce (vlevo) pro dvourozměrnou konvoluci konvoluční maskou  $h(\cdot)$  o rozměrech  $M_h$  a  $N_h = 3 \times 3$ .

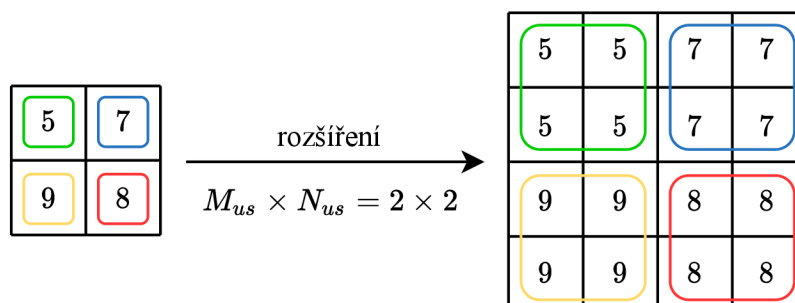


Obr. 8: Ukázka prvního kroku diskrétní dvourozměrné konvoluce konvoluční vrstvy rozšířené zero-paddingem (viz obr. 7) na rozměry  $6 \times 6$  konvoluční maskou  $3 \times 3$ .

Vrstvy sjednocení (pooling) v konvolučních neuronových sítích zmenšují rozměry obrazové funkce shlukováním skupiny pixelů do jednoho pixelu ve výstupní mapě [18]. Obrazová funkce  $f(\cdot)$  je rozdělena do mřížky tak, aby vytvořila skupiny pixelů o rozměrech  $M_p \times N_p$ . Existuje více typů vrstev sjednocení, například max-pooling a avg-pooling. Vrstva max-pooling vybere z každé skupiny pixelů ohraničených mřížkou vždy ten s nejvyšší hodnotou a zapíše ji do výstupní mapy, jak je znázorněno na obr. 9 nahoře. Vrstva avg-pooling zapíše na odpovídající místo v mapě aritmetický průměr všech hodnot skupiny pixelů (viz obr. 9 dole). Opakem vrstvy sjednocení je vrstva rozšíření (up-sample). Ta pro každý pixel vstupní obrazové funkce  $f(\cdot)$  vytvoří ve výstupní mapě skupinu pixelů o rozměrech  $M_{us} \times N_{us}$  se stejnými hodnotami, kterou má pixel ze vstupní funkce (viz obr. 10) [18]. Skupiny pixelů jsou ve výstupní mapě umístěné stejně jako jim odpovídající vstupní pixely.



Obr. 9: Ukázka dvou variant vrstvy sjednocení. Vlevo je původní obrazová funkce o rozměrech  $M \times N = 4 \times 4$ , vpravo jsou obrazové funkce po sjednocení max-pooling mřížkou  $M_p \times N_p = 2 \times 2$  a average-pooling mřížkou  $M_p \times N_p = 2 \times 2$ . Barevně jsou vyznačeny skupiny sjednocených pixelů.



Obr. 10: Ukázka vrstvy rozšíření. Vlevo je původní obrazová funkce o rozměrech  $M \times N = 2 \times 2$ , vpravo je obrazová funkce po rozšíření skupinou pixelů o rozměrech  $M_{us} \times N_{us} = 2 \times 2$ . Barevně jsou vyznačeny skupiny rozšířených pixelů.

Vrstva s úplným propojením se běžně umísťuje před poslední výstupní vrstvou konvoluční neuronové sítě. U konvolučních vrstev, vrstev sjednocení a rozšíření nebyly navzájem propojeny všechny neurony (pixely) vrstvy předcházející se všemi neurony vrstvy následující, protože konvoluční masky a skupiny pixelů vrstev sjednocení a rozšíření nepokrývají celou síť najednou. Změna hodnoty určitého neuronu vrstvy předcházející nemusí mít vliv na změnu hodnoty všech neuronů ve vrstvě následující. Vrstva s úplným propojením propojí všechny neurony vrstvy předcházející se všemi neurony vrstvy následující [18].

### 3.4.3 Augmentace

Pro účinné trénování všech hlubokých sítí je potřeba obsáhnout velké množství různorodých vzorků v trénovací sadě. Počet i různorodost lze navýšit augmentací. Augmentované vzorky vzniknou obrazovou transformací vzorků původních [21]. Sítě, které jsou natrénovány na augmentovaných datech, bývají robustnější [21]. Nejčastější typy augmentací jsou geometrické a fotometrické.

#### Geometrické transformace

Geometrická augmentace vytváří syntetický vzorek geometrickou transformací souřadnic  $(x, y)$  obrazové funkce  $f(\cdot)$  na souřadnice

$$(x', y') = \mathcal{T}[(x, y)], \quad (15)$$

kde  $(x', y')$  jsou souřadnice výsledného augmentovaného obrazu a  $\mathcal{T}(\cdot)$  je transformační funkce. V úlohách strojového vidění se však používá maticový zápis této transformace

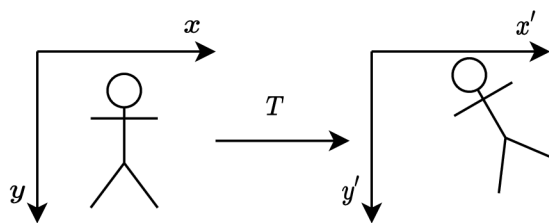
$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} T, \quad (16)$$

kde  $T$  je matice afinní transformace [18]. Obecně se dá říct, že geometrická transformace převádí jeden souřadný systém do druhého.

Podoba transformační matice  $T$  se mění v závislosti na typu transformace. Transformace rotace (rotation) znázorněná na obr. 11 s maticí afinní transformace

$$T = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (17)$$

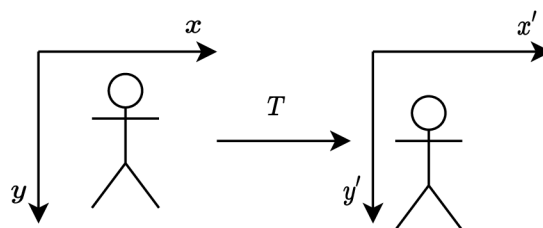
rotuje pixely kolem středu souřadného systému o úhel  $\theta$ . Kladný úhel  $\theta$  rotuje s obrazem proti směru hodinových ručiček.

Obr. 11: Ukázka rotace o kladný úhel  $\theta$ .

Transformace translace (translation) na obr. 12 posouvá obraz v rovině souřadného systému ve směru osy  $x$  a osy  $y$ . Její transformační matice je definována jako

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}, \quad (18)$$

kde  $t_x$  je počet pixelů, o který se má obraz posunout ve směru osy  $x$ , a  $t_y$  je počet pixelů, o který se obraz má posunout ve směru osy  $y$ .

Obr. 12: Ukázka translace o zápornou hodnotu  $t_x$  a kladnou hodnotu  $t_y$ .

Dalším typem afinní transformace je převrácení (flipping). Obraz se převrací horizontálně, nebo vertikálně. Ukázka horizontálního převrácení obrazu je na obr. 13. Matice afinní transformace pro horizontální převrácení je definována jako

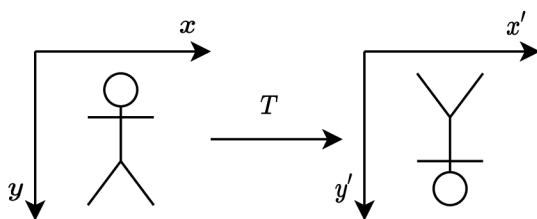
$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & N & 1 \end{bmatrix}, \quad (19)$$

a pro vertikální převrácení je

$$T = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ M & 0 & 1 \end{bmatrix}. \quad (20)$$

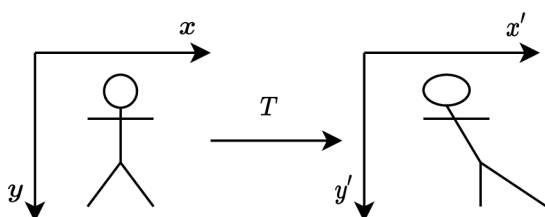
Zkosení (shearing) obrazu, které je ukázáno na obr. 14, se realizuje pomocí matice afinní transformace

$$T = \begin{bmatrix} 1 & s_y & 0 \\ s_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (21)$$



Obr. 13: Ukázka horizontálního převrácení.

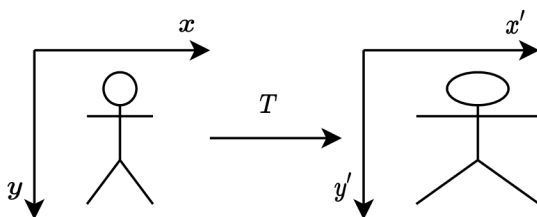
kde  $s_x$  je poměr zkosení obrazu ve směru osy  $x$  a  $s_y$  je poměr zkosení obrazu ve směru osy  $y$ .

Obr. 14: Ukázka zkosení o kladný parametr  $s_x$ .

Poslední uváděnou afinní transformací je škálování (scaling). Její matice  $T$  má podobu

$$T = \begin{bmatrix} c_y & 0 & 0 \\ 0 & c_x & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (22)$$

kde  $c_x$  je škálování ve směru osy  $x$  a  $c_y$  je škálování ve směru osy  $y$ . Ukázka škálování je na obr. 15.

Obr. 15: Ukázka geometrické transformace škálování o kladný parametr  $c_x$ .

### Fotometrické transformace

Zatímco geometrické augmentace mění polohy jednotlivých pixelů, tedy je převádějí do jiného souřadného systému, šumy, což jsou speciální typy fotometrických transformací, mění hodnoty pixelů [22]. Změna každého pixelu na jinou hodnotu je dána pravděpodobnostním modelem.



Mezi základní typ šumu patří gaussovský šum

$$P(f(x, y)) = \sqrt{\frac{1}{2\pi\sigma^2}} e^{-\frac{(f(x, y) - q)^2}{2\sigma^2}}, \quad (23)$$

kde  $P(\cdot)$  je funkce hustoty pravděpodobnosti změny hodnoty pixelu,  $q$  je střední hodnota Gaussova rozdělení pravděpodobnosti a  $\sigma$  je směrodatná odchylka Gaussova rozdělení pravděpodobnosti. Šum speckle je odvozený od gaussova šumu, jeho podoba je k nahlédnutí v [22, 23]. Šum sůl a pepř mění hodnoty některých pixelů na 0 (černá), nebo 255 (bílá). Jeho pravděpodobnostní funkce je definována jako

$$P(f(x, y)) = \begin{cases} P_s, & \text{pro } f(x, y) = 255, \\ P_p, & \text{pro } f(x, y) = 0, \\ 0, & \text{jinak,} \end{cases} \quad (24)$$

kde  $P_s$  je pravděpodobnost zašumění daného pixelu solí tedy změna jeho hodnoty na 255 a  $P_p$  je zašumění pepřem, což znamená změnu hodnoty pixelu na 0 [22]. Všechny zmíněné šумы jsou ukázány na obr. 16.



Obr. 16: Ukázka šumů. Gaussovský šum s hodnotami  $q = 0$  a  $\sigma = 20$ . Speckle s hodnotami  $q = 0$  a  $\sigma = 1$ . Sůl a pepř s hodnotami  $P_s = 0,05$  a  $P_p = 0,05$ . Fotografie na obrázku byly převzaty a upraveny z [24].

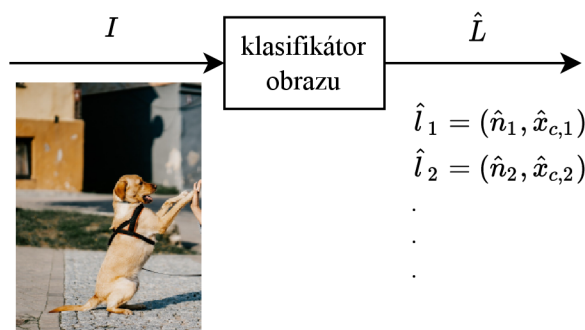
### 3.5 Klasifikace obrazu

Klasifikace obrazu je úloha strojového vidění. Klasifikátor obrazu je většinou konvoluční neuronová síť, která slouží k rozpoznání předmětu v obraze [25]. Obrazu přiřazuje určitou třídu  $n$ . Třída je souhrnné označení pro určitý typ obrazových dat, který má podobné rysy (kočka, pes, slon apod.). Nevýhodou této metody je, že v obraze nerozpozná více předmětů.

Vstupní vrstva klasifikátoru obrazu má stejné rozměry jako obrazová funkce  $f(\cdot)$  vstupního obrazu  $I$  a nabývá i stejných hodnot. Architektura sítě obvykle obsahuje konvoluční vrstvy, vrstvy sjednocení a vrstvy s úplným propojením. Počet neuronů výstupní vrstvy odpovídá počtu tříd, které je klasifikátor schopný rozeznat. Množina všech možných tříd se označuje jako  $X$ , kde  $n \in X$ . Hodnoty neuronů v poslední vrstvě určují odhad jistoty  $\hat{x}_c$  (confidence), s jakou klasifikátor přiřadí  $I$  jednotlivým třídám. Čím vyšší hodnotu  $\hat{x}_c$  má neuron poslední vrstvy reprezentující určitou třídu, tím je pravděpodobnější, že vstupní obraz  $I$  odpovídá této třídě. Klasifikátor vrací na výstupu množinu  $\hat{L}$ , která obsahuje uspořádané dvojice

$$\hat{l} = (\hat{n}, \hat{x}_c), \quad (25)$$

kde  $\hat{n}$  je odhadovaná třída a  $\hat{x}_c$  je odhad jistoty. Na obr. 17 je ukázáno, jak klasifikátor funguje.



Obr. 17: Ukázka klasifikátoru, který klasifikuje vstupní obraz  $I$  třídy pes. Fotografie na obrázku byla převzata a upravena z [24].

Trénování klasifikátorů probíhá formou učení s učitelem. Očekávané výstupy, které se používají jako zpětná vazba během trénování a jsou přiřazené ke každému trénovacímu vzorku, tvoří množinu  $L$  uspořádaných dvojic

$$l = (n, x_c), \quad (26)$$

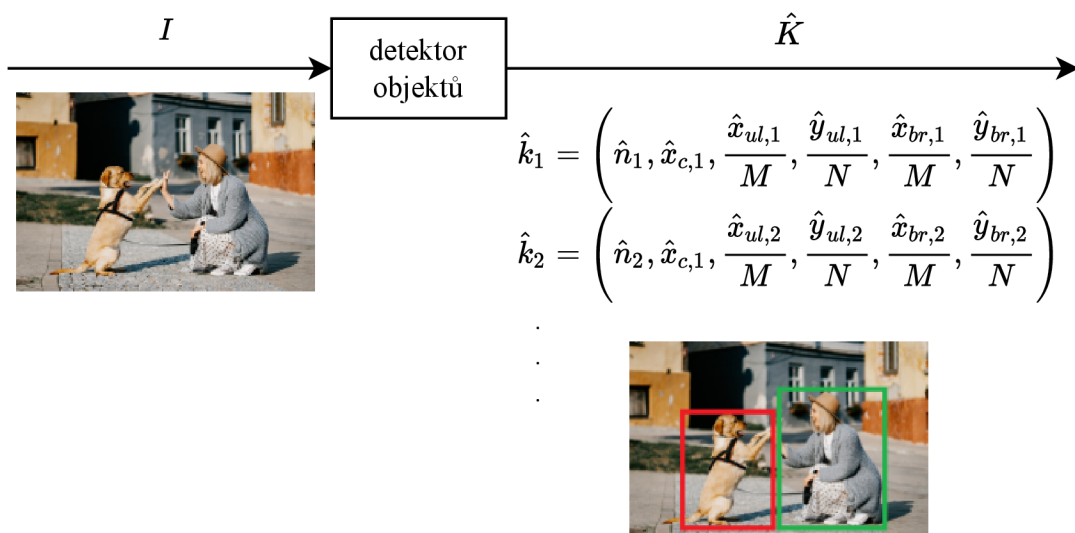
kde  $x_c$  je očekávaná jistota. Hodnota  $x_c$  nabývá pouze 0 (obraz  $I$  nenáleží této třídě) nebo 1 (obraz  $I$  náleží této třídě). Procesu označování vzorků trénovací sady očekávanými výstupy se říká štítkování.

### 3.6 Detekce objektů

Detekce objektů v obraze je úloha strojového vidění založená na konvolučních neuronových sítích [26]. V jednom obraze  $I$  se vyhledávají bounding-boxy (pozice) objektů, které jsou si podobné v určitých rysech [27] (tvar, barva, prostředí výskytu apod.) a rozřazují se do tříd  $n$  z množiny  $X$ . Bounding-box obsahuje odhad souřadnic levého horního a pravého spodního bodu, které definují obdélníkovou podoblast nalezeného objektu v obrazové funkci  $f(\cdot)$ . Do detektoru objektů vstupuje obraz  $I$  a vystupuje z něj množina  $\hat{K}$ , která obsahuje uspořádané šesticice

$$\hat{k} = \left( \hat{n}, \hat{x}_c, \frac{\hat{x}_{ul}}{M}, \frac{\hat{y}_{ul}}{N}, \frac{\hat{x}_{br}}{M}, \frac{\hat{y}_{br}}{N} \right), \quad (27)$$

kde  $\hat{x}_{ul}$  a  $\hat{y}_{ul}$  je odhad souřadnic levého horního rohu bounding-boxu,  $\hat{x}_{br}$  a  $\hat{y}_{br}$  je odhad souřadnic pravého spodního rohu bounding-boxu pro každý nalezený objekt v obraze. Ukázka detektoru objektů je na obr. 18.



Obr. 18: Ukázka detektoru objektů. Ve vstupním obraze  $I$  jsou detekovány objekty třídy pes (červený bounding-box) a člověk (zelený bounding-box). Fotografie byly převzaty a upraveny z [24].

Stejně tak jako trénování klasifikátorů, i trénování detektorů objektů probíhá formou učení s učitelem. Každému vzorku se přiřazuje množina  $K$ , která obsahuje uspořádané šesticice

$$k = \left( n, x_c, \frac{x_{ul}}{M}, \frac{y_{ul}}{N}, \frac{x_{br}}{M}, \frac{y_{br}}{N} \right), \quad (28)$$

kde  $x_{ul}$  a  $y_{ul}$  jsou očekávané souřadnice levého horního rohu bounding-boxu a  $x_{br}$  a  $y_{br}$  jsou očekávané souřadnice pravého spodního rohu bounding-boxu. Před trénováním se trénovací sada štítkuje očekávanými bounding-boxy.

Nejpoužívanější ztrátovou funkcí pro detekci objektů je Intersection over Union (IoU), která je daná vztahem

$$C_{IoU} = \frac{A_p \cap A_{gt}}{A_p \cup A_{gt}}, \quad (29)$$

kde  $C_{IoU}(\cdot)$  je ztrátová funkce IoU,  $A_p$  je plocha plocha výstupního bounding boxu detektoru objektů a  $A_{gt}$  je plocha očekávaného bounding-boxu (ground truth) ve vstupním obraze  $I$  [28].

### 3.6.1 mean Average Precision

Pro evaluaci detektoru objektů se nejčastěji používá metrika mean Average Precision (mAP). Pro její definování je potřeba zavést několik pojmů.

Mějme prahovou hodnotu  $t_{IoU}$ . Detekce je pravdivě pozitivní (TP), pokud  $C_{IoU} > t_{IoU}$ , nepravdivě pozitivní (FP), pokud  $C_{IoU} < t_{IoU}$ , nebo nepravdivě negativní (FN), pokud objekt nebyl detekován vůbec. Přesnost  $P$  je pak

$$P = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|}, \quad (30)$$

kde  $|\cdot|$  je počet prvků. Recall  $R$  je dán vztahem

$$R = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|}. \quad (31)$$

Hodnoty  $P$  a  $R$  se spočítají pro všechny evaluační vzorky pro řadu prahových hodnot  $t_{IoU}$ . Počet počítaných prahových hodnot  $t_{IoU}$  je  $J_{PR}$ , kde  $0 > t_{IoU} > 1$  a hodnoty  $t_{IoU}$  jsou na tomto intervalu rovnoměrně rozložené. Vznikne tak funkční závislost přesnosti  $P$  na recallu  $R$ , které se v odborné literatuře říká křivka precision-recall [29]. Plocha pod touto křivkou je hodnota average precision (AP)

$$P_A = \sum_{i=2}^{J_{AP}} (R_i - R_{i-1}) P_i, \quad (32)$$

kde  $P_A$  je hodnota AP,  $R_i$  a  $P_i$  jsou hodnoty  $i$ -té prahové hodnoty  $t_{IoU}$  na intervalu křivky precision-recall. Metrika mAP je aritmetický průměr hodnot  $P_A$  pro všechny třídy  $n$

$$P_{mA} = \frac{\sum_{n \in X} AP_n}{|X|}, \quad (33)$$

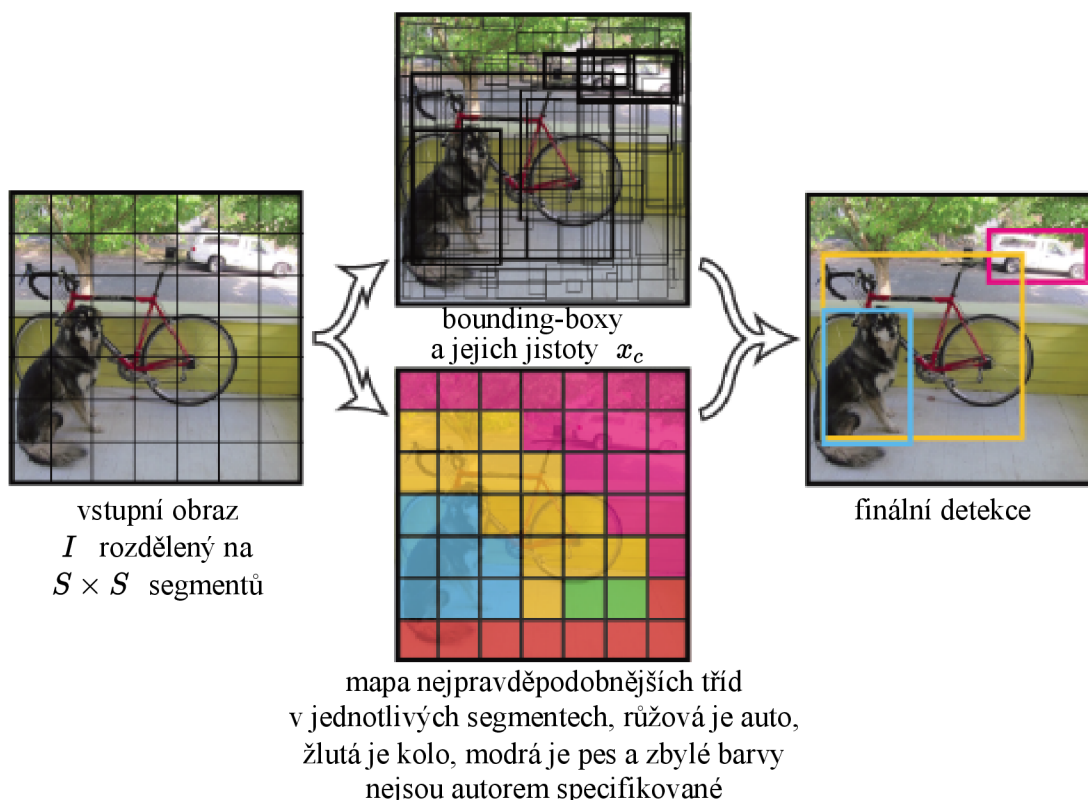
kde  $P_{mA}$  je hodnota mAP,  $AP_n$  je hodnota AP třídy  $n$  [28, 29].

### 3.6.2 YOLO

Existuje mnoho různých detektorů objektů. Liší se ve způsobu zpracování dat, používanými ztrátovými funkcemi apod. Obecně dělíme detektory objektů na jedno-fázové (one-stage) a dvoufázové (two-stage) [26]. Dvoufázové detektory objektů nejprve vytvoří mapu podobných oblastí, pomocí které pak detekují objekty v obraze.

Mezi nejvýznamnější detektory objektů patří YOLO (one-stage) a R-CNN (two-stage). R-CNN (Region-based Convolutional Neural Network) poprvé popsal Girshick v roce 2014 [30]. Fast R-CNN [31] a Faster R-CNN [32] využívají pro tvorbu mapy podobných oblastí konvoluční neuronové sítě a tím algoritmus urychlují. Rodina R-CNN je obecně považovaná za pomalé algoritmy [26].

YOLO (You Only Look Once) poprvé představil Redmon a kolektiv v roce 2016 [33]. Tento detektor je jednofázový, proto nevytváří mapy podobných oblastí, díky čemuž je mnohem rychlejší. Algoritmus rozdělí vstupní obraz mřížkou na  $S \times S$  segmentů (viz obr. 19 vlevo). Pro každý segment vrací YOLO na výstupu informaci, zda v něm pomocí klasifikátoru obrazu našel nějaký objekt, jaké je třídy (viz obr. 19 dole), s jakou jistotou  $\hat{x}_c$  a jaké má souřadnice jeho bounding-box. Objekt může přesahovat i do sousedních segmentů. Na obr. 19 nahoře je ukázáno, že jeden objekt může být detekovaný vícekrát. YOLO spočítá hodnotu IoU pro všechny bounding-boxy překrývající se tříd. Následně odstraní všechny bounding-boxy, které nepřekročí prahovou hodnotu  $t_{bb}$  svou hodnotou  $\hat{x}_c$  (viz obr. 19 vpravo). Tato operace se nazývá non-maxima suppression. YOLO může v každém segmentu detekovat více objektů.

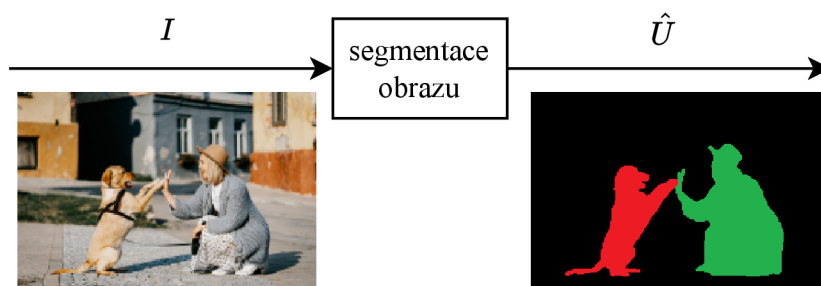


Obr. 19: Schéma procesů detektoru YOLO. Nalevo je vstupní obraz rozdělený na segmenty, napravo jsou v obrazu vyznačené detekované bounding-boxy po non-maxima suppression. Převzato a upraveno z [34].

YOLO byl postupně vyvíjen až do čtvrté verze YOLOv4, která je optimalizovaná pro paralelní výpočty, je rychlejší a přesnější než verze předchozí. Byla představena Bochkovskiyem v roce 2020 [34]. Nejnovější verze YOLOv4 se liší od ostatních především architekturou konvolučních neuronových sítí.

### 3.7 Segmentace obrazu

Segmentace obrazu je metoda strojového vidění, která rozděljuje pixely vstupního obrazu  $I$  do segmentů tříd  $n$  [35]. Výstupem segmentace obrazu je množina  $\hat{U}$  odhadovaných map  $\hat{u}$ , které mapují pixely všech tříd v obrazové funkci  $f(\cdot)$  vstupního obrazu  $I$ . Jedné třídě náleží jedna výstupní mapa, existují však i metody segmentace obrazu, které mapují do jedné výstupní mapy více tříd odlišnými hodnotami, jak je to ukázáno na obr. 20.



Obr. 20: Ukázka segmentace obrazu tříd pes a člověk. Napravo jsou dvě výstupní mapy. Červenou barvou je mapována třída pes a zelenou barvou třída člověk. Fotografie na obrázku byla převzata a upravena z [24].

Trénování modelů pro segmentaci obrazu probíhá formou učení s učitelem. Každému vzorku se přiřadí očekávaná množina  $U$  výstupních map  $u$ . Mezi používané ztrátové funkce patří IoU a křížová entropie (CE, cross entropy)

$$E_C = -\frac{1}{|\Phi|} \sum_{i \in \Phi} \sum_{n \in X} z_{i,n} \log(\hat{z}_{i,n}), \quad (34)$$

kde  $E_C$  je hodnota CE,  $\Phi$  je množina všech hodnot pixelů vstupního obrazu  $I$ ,  $z_{i,n}$  je očekávaná hodnota pixelu třídy  $n$  a  $\hat{z}_{i,n}$  je výstupní hodnota pixelu třídy  $n$  [36]. Dále pak binární křížová entropie (BCE, binary cross entropy)

$$E_{BC} = -\frac{1}{|\Phi|} \sum_{i \in \Phi} [z_i \log(\hat{z}_i) + (1 - z_i) \log(1 - \hat{z}_i)], \quad (35)$$

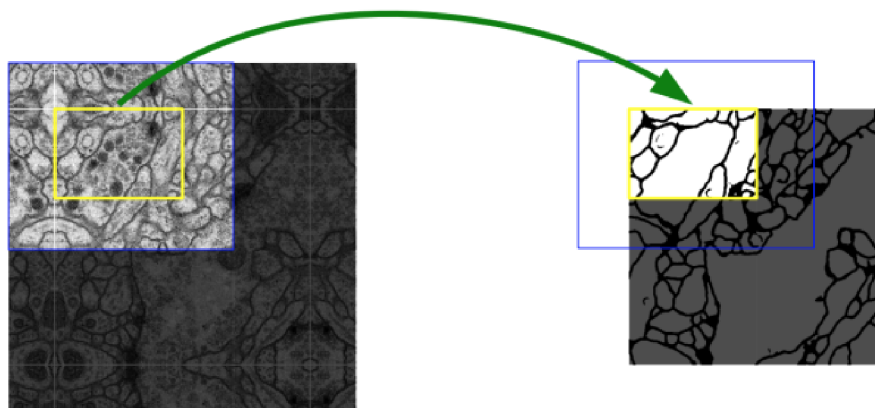
kde  $E_{BC}$  je hodnota BCE [36]. BCE je speciální případ CE, kde  $|X| = 2$ . Pro evaluaci se používají různé funkce, jako jsou například CE, BCE nebo mAP.

Existuje mnoho modelů s různými architekturami pro segmentaci obrazu. Téměř všechny jsou založené na hlubokých konvolučních sítích [35]. Nejpoužívanější

z nich jsou rozdělené na část enkodér a dekodér. Tyto části jsou řazeny za sebou. Enkodér zmenšuje rozměry konvolučních vrstev a zvyšuje počty konvolučních masek, dekodér pak rozměry zpět zvětšuje a redukuje počet masek. Mezi nejznámější architektury patří SegNet [37], V-Net pro 3D segmentaci [38] a U-Net [39].

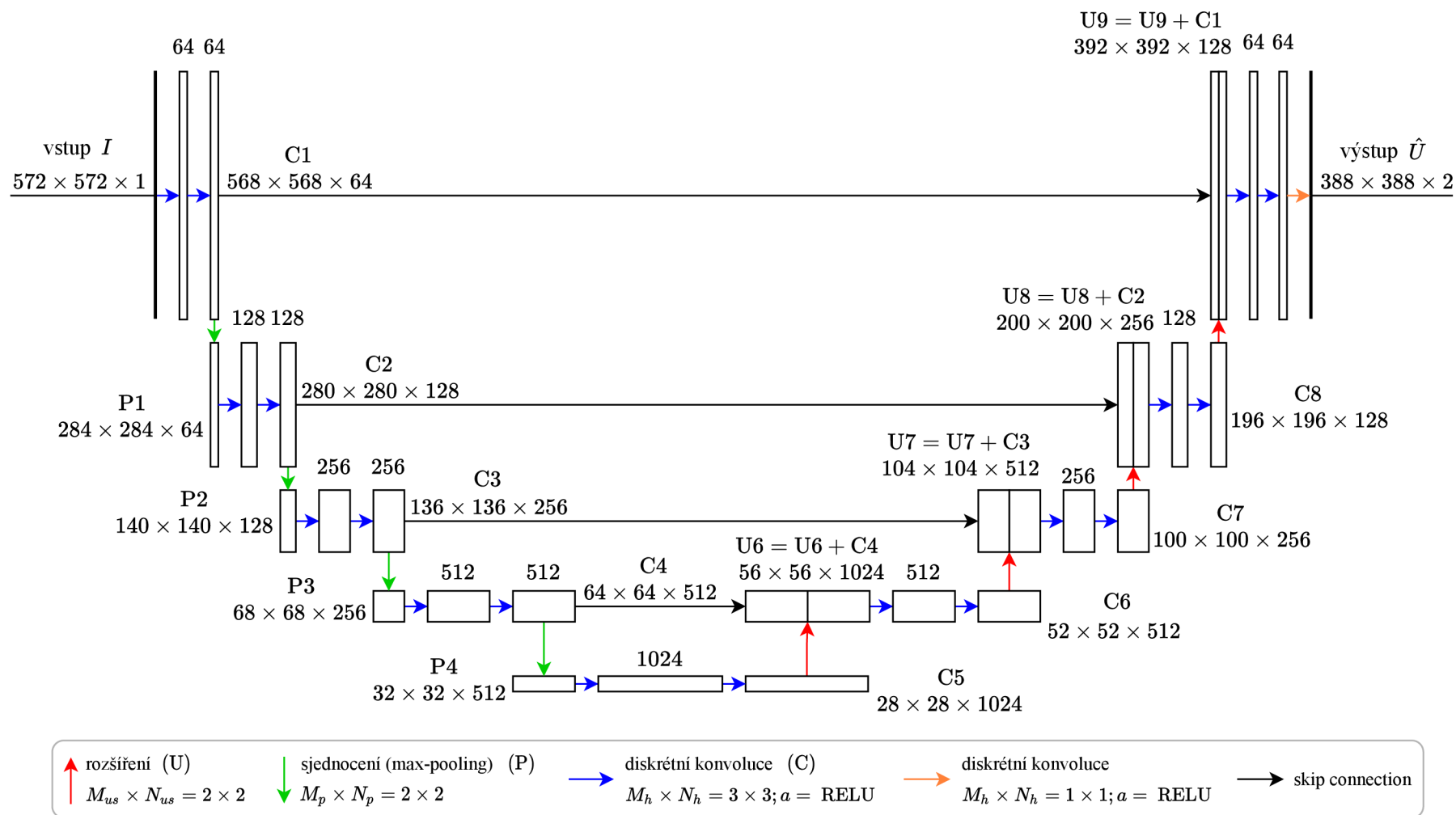
### 3.7.1 U-Net

U-Net poprvé popsal Ronneberger a kolektiv v roce 2015 [39]. Jedná se o jednu z nejefektivnějších metod segmentace obrazu. U-Net funguje na principu hluboké konvoluční sítě. Původní účel U-Netu byl segmentovat ve fotografiích z mikroskopu určité tkáně v lidském mozku. Na obr. 21 je ukázána segmentace biologických buněk.



Obr. 21: Ukázka segmentace obrazu pomocí sítě U-Net. Vpravo je vstupní obraz, vlevo je výstupní mapa třídy buňka, na které jsou bílou barvou namapovány buňky. Černá barva jsou místa, kde U-Net buňky nedetekoval. [39]

Obr. 22 popisuje architekturu sítě U-Net tak, jak ji definoval Ronneberger [39]. Vstupní obraz  $I$  je přijímán dvěma konvolučními vrstvami (C). Autor původní architektury nepoužil zero-padding, a proto mají výstupní mapy za každou konvoluční vrstvou menší rozměry. Následně jsou výstupní mapy sjednocovány (P). Vstupní obraz tímto prošel jednou úrovní sítě U-Net. V další úrovni se celý proces opakuje, ovšem vždy s dvojnásobným počtem konvolučních masek v jedné vrstvě, jak je ukázáno na obr. 22. U-Net na obr. 22 má definovaných pět úrovní. Na poslední úrovni se výstupní mapa konvoluční vrstvy nesjednocuje, ale začíná se rozšiřovat (dojde tedy ke snížení úrovně) (U) a k výsledku je nakopírována jedna konvoluční vrstva, která odpovídá stejné úrovni (viz obr. 22). Této operaci se říká skip connection. Výsledek je přijímán dvěma konvolučními vrstvami s dvakrát nižším počtem konvolučních masek než v předchozím kroku. Jakmile proběhne tento proces tolikrát, kolik má síť U-Net definovaných úrovní, na výstupní mapu poslední konvoluční vrstvy se aplikuje konvoluční vrstva, která má právě tolik konvolučních masek, kolik síť U-Net detekuje tříd  $n$ . Výstupem je množina odhadovaných výstupních map  $\hat{U}$ .



Obr. 22: Architektura sítě U-Net o pěti úrovních a 64 konvolučními maskami v konvolučních vrstvách první úrovně. Síť přijímá černobílou fotografii o rozměrech  $572 \times 572 \times 1$  pixelů a vrací dvě mapy  $\hat{u}$  pro každou třídu  $n$  o rozměrech  $388 \times 388 \times 2$  pixelů. Převzato a upraveno z [39].

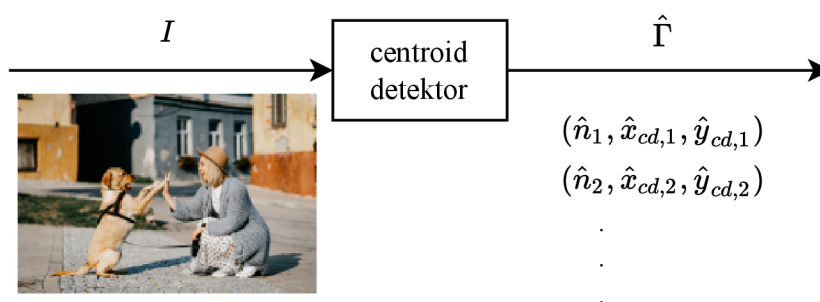


### 3.8 Centroid detektor

Centroid detektor je zvláštní typ detektoru objektů, který nedetekuje bounding-boxy, ale pozice středů objektů [40]. Nebylo by sice náročné přepočítat souřadnice výstupních bounding-boxů klasického detektoru objektů na středy, v některých aplikacích pro počítání velmi malých objektů, které jsou blízko u sebe, by to však nebylo účinné [7]. Centroid detektor na vstupu přijímá vstupní obraz  $I$  a vrací množinu  $\hat{\Gamma}$ , pro kterou platí:

$$(\hat{n}, \hat{x}_{cd}, \hat{y}_{cd}) \in \hat{\Gamma}, \quad (36)$$

kde  $\hat{x}_{cd}$  a  $\hat{y}_{cd}$  jsou odhady souřadnic středu objektu třídy  $\hat{n}$ , jak je ukázáno na obr. 23.



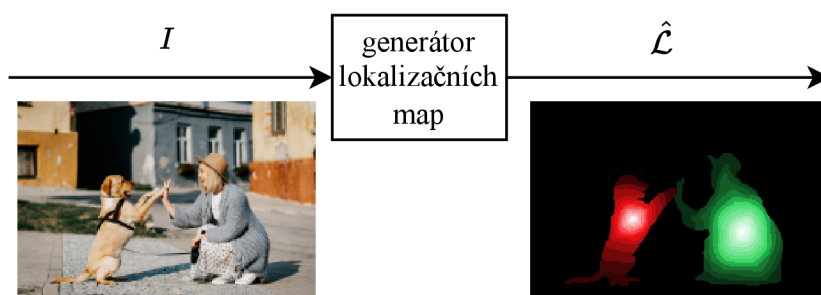
Obr. 23: Ukázka Centroid detektoru, který přijímá vstupní obraz  $I$  a vrací množinu  $\hat{\Gamma}$ . Fotografie na obrázku byla převzata a upravena z [24].

V podkapitole 3.1 bylo zjištěno, že vhodným způsobem pro detekci homogenních objektů je nejprve provést jejich segmentaci a výstup zpracovat vhodným algoritmem podle typu řešeného problému. Centroid detektor je vhodný pro detekci polohy a počtu homogenních objektů, které jsou velmi blízko u sebe, protože je rozdělený právě na část segmentační (generátor lokalizačních map) a část, která v lokalizačních mapách detekuje středy objektů (Centroid counterpoint).

#### 3.8.1 Generátor lokalizačních map

Generátor lokalizačních map je konvoluční neuronová síť založená na architektuře U-net. U-Net vrací mapy, ve kterých jsou třídy mapovány jednou hodnotou. Docent Doležel provedl experiment, ve kterém trénoval síť U-Net na mapách s hodnotami 0 až 255 [40]. Hodnoty v mapách byly štítkovány tak, aby měl každý štítek ve svém středu hodnotu 255 a do stran od něj hodnoty postupně klesaly až na nulu. Výstupem jsou tedy namísto skvrn s jednou hodnotou pro každou třídu rozostřené gradienty s maximem uprostřed. Doleželovi se experiment povedl. Generátor lokalizačních map má za cíl zjednodušit počítání detekovaných objektů tím, že zabrání jejich splývání.

Generátor lokalizačních map přijímá vstupní obraz  $I$  a vrací odhad lokalizační mapy  $\hat{\mathcal{L}}$ , jak je ukázáno na obr. 24. Pro každou třídu vrací jednu mapu.

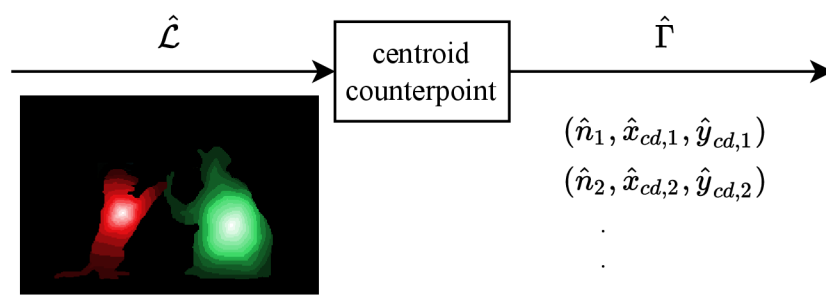


Obr. 24: Ukázka generátoru lokalizačních map, který vytváří odhad lokalizační mapy  $\hat{\mathcal{L}}$  s rozostřeným gradientem uprostřed dvou objektů. Lokalizační mapy různých tříd jsou spojeny a barevně rozlišeny. Červený gradient lokalizuje střed objektu třídy pes a zelený gradient lokalizuje střed objektu třídy člověk. Fotografie na obrázku byla převzata a upravena z [24].

Trénování generátoru lokalizačních map probíhá formou učení s učitelem. Vzorkům se přiřazuje očekávaná lokalizační mapa  $\mathcal{L}$ . Štítkování lokalizačních map  $\mathcal{L}$  probíhá vykreslováním gradientů do míst s očekávanými středy objektů množiny  $\Gamma$ . Jako ztrátová funkce byla použita binární křížová entropie (BCE).

### 3.8.2 Centroid counterpoint

Centroid counterpoint je druhá část Centroid detektoru, která má za úkol lokalizovat polohy lokálních maxim gradientů v lokalizačních mapách. Gradienty nemusí být homogenní, mohou mít různou velikost, různou hodnotu maxima a nepravidelný tvar. Na vstupu přijímá odhad lokalizační mapy  $\hat{\mathcal{L}}$  a na výstupu vrací množinu  $\hat{\Gamma}$ . Chod Centroid counterpointu je znázorněn na obr. 25.



Obr. 25: Ukázka chodu Centroid counterpointu.

Všechny hodnoty pixelů v lokalizační mapě jsou lineárně interpolovány tak, aby nejvyšší hodnota byla 255. Hodnoty jsou pak zaokrouhleny dolů na celá čísla.

Takto upravená mapa je připravena pro další zpracování. Matematickými operacemi se vytváří pomocné obrazové funkce [40]. Poslední funkce vytváří množinu  $\hat{\Gamma}$ .

První operací je maxfiltr

$$B(x, y, n) = \max_{(s,t) \in S_{xy}} \{ \hat{\mathcal{L}}(s, t, n) \}, \quad (37)$$

kde  $S_{xy}$  je soubor prostorových souřadnic v masce maxfiltru o rozměrech  $M_B \times N_B$  se středem v bodě  $(x, y)$ . Maxfiltr  $B(\cdot)$  zvýrazní nenulová místa v lokalizační mapě  $\hat{\mathcal{L}}$  a přitom nezmění hodnoty lokálních maxim. Následně se vytvoří maska

$$B_1(x, y, n) = \begin{cases} 255, & \text{pokud } B(x, y, n) = \hat{\mathcal{L}}(s, t, n), \\ 0, & \text{jinak,} \end{cases} \quad (38)$$

ve které jsou mapována místa, kde mají  $B(\cdot)$  a  $\hat{\mathcal{L}}$  stejné hodnoty. Pixely s hodnotou 255 mohou být středy gradientů, v  $B_1(\cdot)$  se však může nacházet i spousta dalších nenulových pixelů mimo místa s gradienty. K jejich odfiltrování slouží maska

$$\Omega(x, y, n) = \begin{cases} 255, & \text{pokud } \hat{\mathcal{L}}(s, t, n) = 0, \\ 0, & \text{jinak.} \end{cases} \quad (39)$$

Všechna místa s nulovou hodnotou v původním obrazu se vyznačí do masky  $\Omega(\cdot)$ . Maska

$$\Omega_{\ominus}(x, y, n) = \min_{(s,t) \in \varepsilon} \{ \Omega(x + s, y + t, n) \}, \quad (40)$$

kde  $\varepsilon$  je okolí eroze o nastavitelné velikosti  $M_{\varepsilon} \times N_{\varepsilon}$ , je eroze masky  $\Omega(\cdot)$ . Maska  $\Omega_{\ominus}(\cdot)$  mapuje pixely, které můžeme vyloučit z potenciálních středů gradientů lokálních maxim. V masce

$$B_{\Omega}(x, y, n) = B_1(x, y, n) \oplus \Omega_{\ominus}(x, y, n) \quad (41)$$

je logický XOR masky  $B_1(\cdot)$  a masky  $\Omega_{\ominus}(\cdot)$ . Hledané středy gradientů mají hodnotu 255 v mapě

$$\Xi(x, y, n) = \begin{cases} 255, & \text{pokud } B_{\Omega}(x, y, n) = 1 \wedge \hat{\mathcal{L}}(s, t, n) \geq t_m, \\ 0, & \text{jinak,} \end{cases} \quad (42)$$

kde  $t_m$  je práh citlivosti,  $x \in X_{\Xi}$ ,  $y \in Y_{\Xi}$ ,  $n \in N_{\Xi}$ ,  $X_{\Xi} = \{0, \dots, M_{\Xi} - 1\}$ ,  $Y_{\Xi} = \{0, \dots, N_{\Xi} - 1\}$ ,  $N_{\Xi} = \{0, \dots, |X| - 1\}$ ,  $M_{\Xi}$  je šířka a  $\times N_{\Xi}$  je výška mapy  $\Xi$ . Práh citlivosti  $t_m$  byl definován proto, že gradienty v  $\Xi$  nemusí mít ve svém maximu hodnotu 255. Souřadnice poloh nalezených gradientů jsou ve výsledné množině

$$\hat{\Gamma} = \arg \max_{(x,y,n) \in S_{xyn}} \{ \Xi(x, y, n) \}, \quad (43)$$

kde  $S_{xyn} = X_{\Xi} \times Y_{\Xi} \times N_{\Xi}$ .

### 3.9 Souborové učení

Souborové učení se v zahraniční literatuře označuje jako ensemble learning. Princip souborového učení poprvé popsali Dasarathy a Sheela v roce 1979 [41]. Jde o využití několika modelů, které jsou natrénované na stejné trénovací sadě jiným způsobem [42]. Souborové učení lze chápat jako rozhodovací strom, ve kterém se podle předem daného algoritmu rozhoduje, který model bude použitý [43].

Modely se mohou lišit v topologii i parametrech, mohou být vytvořeny na odlišných principech. Různé modely mohou mít odlišné výstupy na stejný vstup. Výhoda souborového učení spočívá v možnosti výběru jednoho nejlepšího výstupu nebo ve sloučení několika výstupů v jeden (například průměrem hodnot). Principy souborového učení jsou hojně využívány pro řešení různých typů problému. Ve strojovém učení je to například oprava chyb, doplnění výsledků, zvýšení přesnosti apod. [44].

## 4 ŘEŠENÍ

Pro výsledné řešení v podobě ovladatelného programu byl navržen systém strojového vidění, který je popsán v této kapitole. Celé řešení bylo vytvořeno v jazyce Python 3.8.10 64-bit.

### 4.1 Popis dostupné datové sady a její analýza

V rámci spolupráce s vědci z ÚBO AV ČR v.v.i. a PdF MUNI bylo poskytnuto 3 564 fotografií ještěrek obecných. Autorem všech fotografií je Ing. Radovan Smolinský, Ph.D., z PdF MUNI [4]. Fotografie se nacházely v adresářích rozříděny podle data a místa pořízení. Fotografie byly ve formátech jpg, psd a CR2.

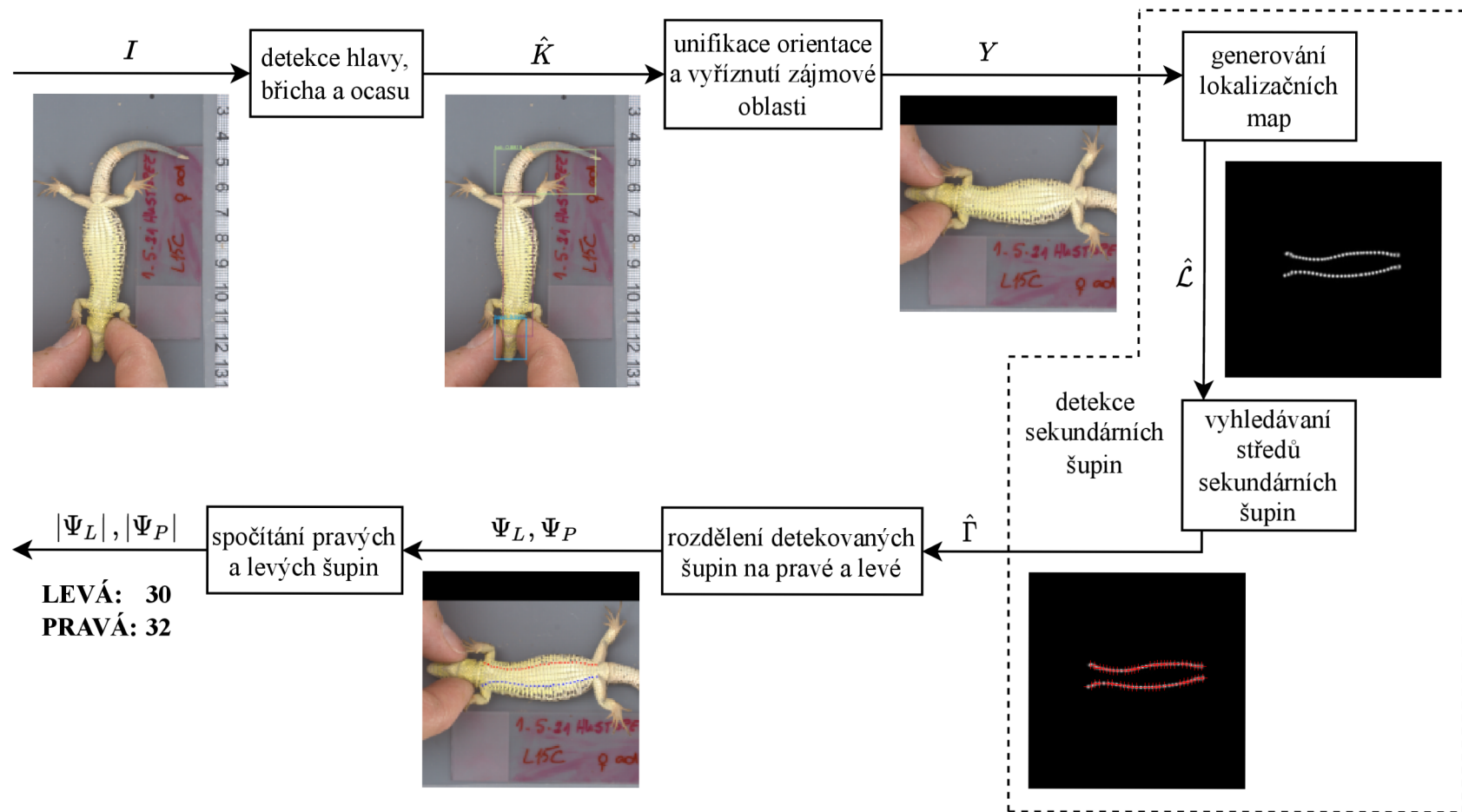
Během fotografování byly ještěrky pokládány na standardní fotografickou tabulku 18 % šedá, díky čemuž jsou dobře viditelné [45]. Snímky byly pořizovány pro minimalizaci variability světelných podmínek ve stínu. I přesto fotografie vykazují jisté známky variability. Databáze fotografií byla vytvářena 4 roky různými fotoaparáty, převážně s Canon EOS 450D. Rozměry fotografií proto nejsou jednotné. Na obr. 26 je příklad fotografie ještěrky.



Obr. 26: Příklad fotografie ještěrky obecné [4].

## 4.2 Popis navrženého řešení

Do navrženého systému strojového vidění vstupují jednotlivé vstupní fotografie  $I$ . Klíčovou metodou navrženého systému jsou konvoluční neuronové sítě. Aby byly výsledky přesnější, tak je potřeba síť trénovat na fotografiích se stejně orientovanými ještěrkami podobné velikosti. Z tohoto důvodu je potřeba vyříznout těla ještěrek v  $I$  a výřezy pak orientovat jedním určitým směrem. Takto připravené výřezy  $Y$  jsou pro trénování vhodnější, protože poskytnuté fotografie jsou různě orientované, mají různé rozměry a velikosti ještěrek. Pro tento úkol byla vybrána síť YOLOv4, a to kvůli její rychlosti a přesnosti oproti ostatním alternativám. Pro detekci šupin ve výřezech  $Y$  byl použitý Centroid detektor, protože díky zvýraznění hledaných šupin pomocí gradientů v lokalizačních mapách  $\hat{\mathcal{L}}$  a až následné detekci pozic  $\hat{\Gamma}$  jejich středů lépe detekuje šupiny, které jsou blízko u sebe. Detekované pozice jsou následně rozdělené na levou a pravou řadu sekundárních šupin  $\Psi_L$  a  $\Psi_P$ . Počty detekovaných pravých a levých šupin  $|\Psi_L|$  a  $|\Psi_P|$  jsou výstupem systému. Na obr. 27 je navržený systém strojového vidění rozdělen do částí. Jednotlivé části budou popsány v následujících podkapitolách.

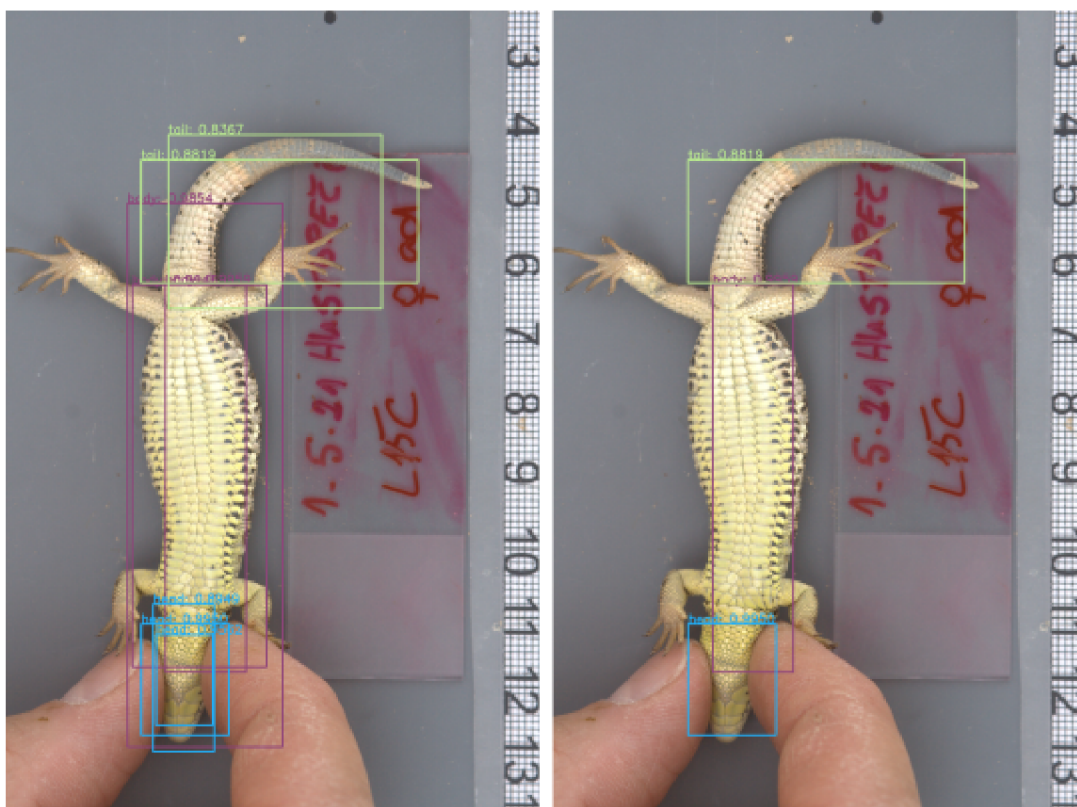


Obr. 27: Znázornění navrženého systému strojového vidění. Šipky zobrazují směr toku dat mezi operacemi v obdélnících. Nad šipkami jsou proměnné reprezentující data, pod šipkami jsou příklady obrázků, které tato data vizualizují. Fotografie na obrázku byly převzaty a upraveny z [4].

#### 4.2.1 Detekce hlavy, břicha a ocasu

Aby byly vstupy pro detektor sekundárních šupin co nejpodobnější (unifikované), je potřeba oříznout a přetočit každou vstupní fotografii  $I$ . Model topologie YOLOv4 byl natrénován na vyhledávání břich, ocasů a hlav na ventrální straně ještěrek. Podrobný popis trénování modelu je v podkapitole 4.4.1.

Systém strojového vidění na vstupu přijme barevnou fotografii o libovolných rozměrech. Rozměry fotografie změni na  $416 \times 416$  pixelů, což bylo ověřeno jako ideální kompromis mezi rychlostí výpočtu a kvalitou fotografie. Rozměry odpovídají rozměrům vstupní vrstvy modelu YOLOv4. Natrénovaný model YOLOv4 vyhledá ve fotografii zmíněné objekty (viz obr. 28 vlevo). Po použití metody non-maxima suppression zbudou jen ty objekty, kterými si je systém nejvíce jistý na dané pozici (viz obr. 28 vpravo). Nastavení systému tak, aby ve výstupní množině  $\hat{K}$  vrátil nejvýše jednu uspořádanou šestici  $\hat{k}$  pro každou třídu  $n$ , bylo inspirováno tvorbou De Boevera [46].



Obr. 28: Výstup YOLOv4. Vlevo jsou zobrazeny detekované objekty před použitím non-maxima suppression a vpravo jsou objekty po použití této metody. Fotografie byly převzaty a upraveny z [4].



### 4.2.2 Unifikace orientace a vyříznutí zájmové oblasti

Pro unifikaci orientace je potřeba, aby byla vyříznuta zájmová oblast (tělo ještěrky) a výřez otočit hlavou na jednu stranu (byla vybrána levá).

Pomocí souřadnic bounding-boxu detekovaného břicha z množiny  $\hat{K}$  je určena kratší strana obdélníku tohoto bounding-boxu. Ta je zvětšena tak, aby měl bounding-box rozměry čtverce (aby byl výřez použitelný v další části systému). Obě strany čtverce jsou zvětšeny parametrem `ENLARGE_BODY` (volí uživatel programu), aby v případě nedokonalé detekce nedocházelo k vyříznutí břicha bez některých šupin. Pokud jsou hrany čtverce za hranami fotografie, je výřez doplněn černou (pixely s hodnotou 0). Celý výřez je převeden na rozměr  $512 \times 512$  pixelů. Tento rozměr odpovídá rozměru vstupní vrstvy detektoru sekundárních šupin.

Bounding-boxy detekovaných objektů z množiny  $\hat{K}$  jsou přepočítány na souřadnice jejich středů. Souřadnice středů břich, ocasů a hlav slouží k unifikaci orientace. Příklad unifikovaného výřezu  $Y$  je na obr. 29. Úhel otočení výřezu je určen podle následujících pravidel:

- V případě, že je detekována hlava i ocas, používá se k určení úhlu pouze ocas (na základě analýzy výsledků bylo zjištěno, že YOLO detekuje přesněji ocas).
- Souřadnice středů detekovaných objektů jsou porovnány v obou osách. Fotografie se otočí tak, aby na ose  $x$  měly detekované středy objektů mezi sebou větší vzdálenost než na ose  $y$ . To zajistí, že řady šupin ještěrky budou rovnoběžné s osou  $x$ . Následně se porovnají nově vzniklé souřadnice středů objektů v ose  $x$  a obraz se otočí o  $180^\circ$ , pokud je střed ocasu detekovaný napravo od středu břicha, nebo střed hlavy nalevo od středu břicha. Díky této operaci bude obraz ještěrky orientovaný hlavou vždy na levou stranu.
- Pokud je detekováno pouze břicho, je úhel natočení určen náhodně tak, aby byl jeho bounding-box orientovaný delší stranou ve směru osy  $x$ . Tato informace je zapsána do výstupního txt souboru.
- Pokud není detekován žádný objekt, systém zapíše informaci do výstupního txt souboru a zpracování fotografie je i pro další části systému strojového vidění ukončeno.



Obr. 29: Unifikovaný výřez  $Y$  vstupní fotografie  $I$  z obr. 26 o rozměrech  $512 \times 512$  pixelů. Velikost a umístění výřezu založena na výstupech YOLOv4 detektoru. Černý pruh v horní části obrazu vznikl doplněním původního obrazu černou. Fotografie byla převzata a upravena z [4].

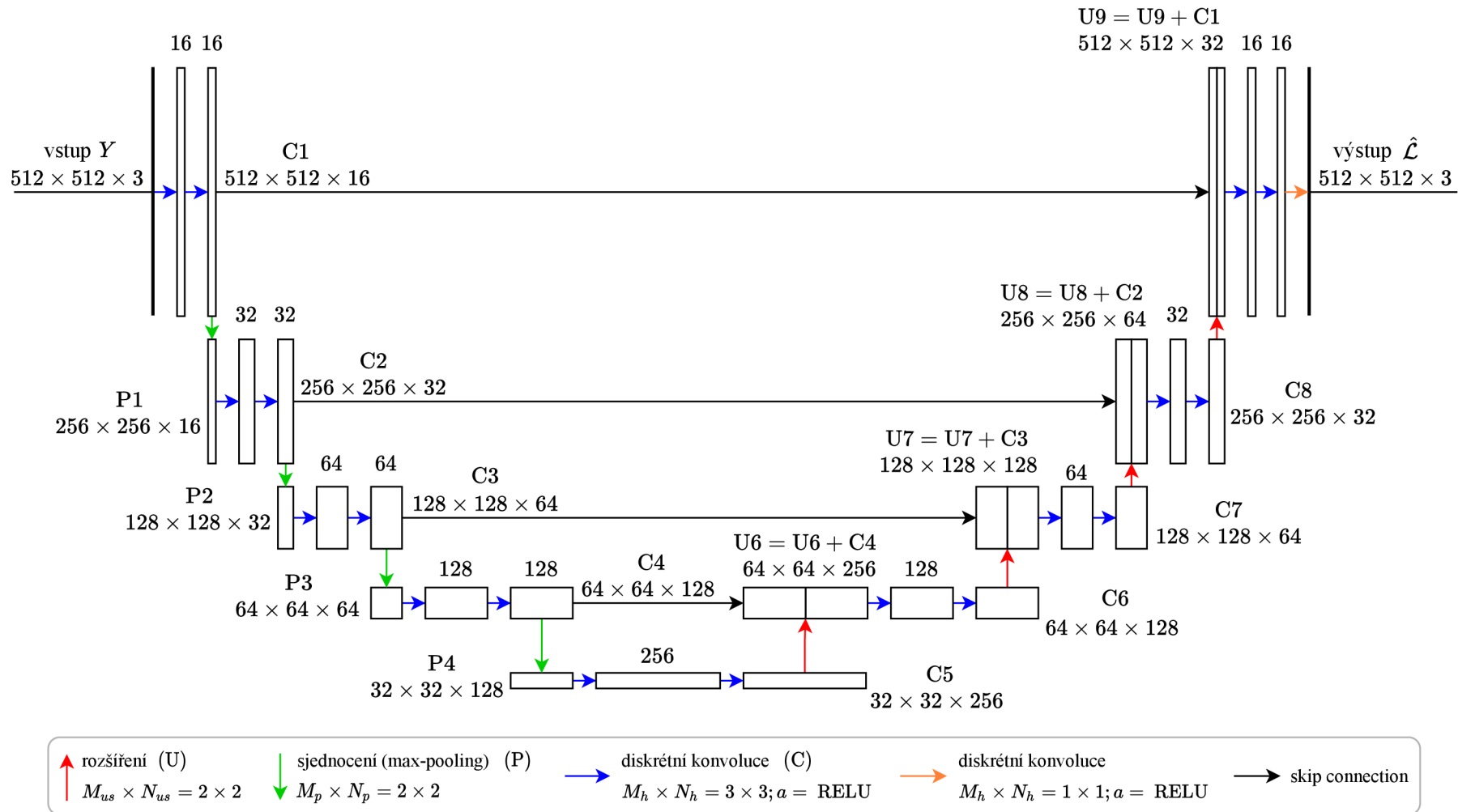
#### 4.2.3 Detekce sekundárních šupin

Pro detekci sekundárních šupin byl použitý Centroid detektor popsáný v podkapitole 3.8, který se skládá z generátoru lokalizačních map a Centroid counterpointu. Vstupem do této části systému strojového vidění je hlavou na levou stranu orientovaný výřez břicha ještěrky  $Y$  o rozměrech  $512 \times 512$  pixelů (viz obr. 29).

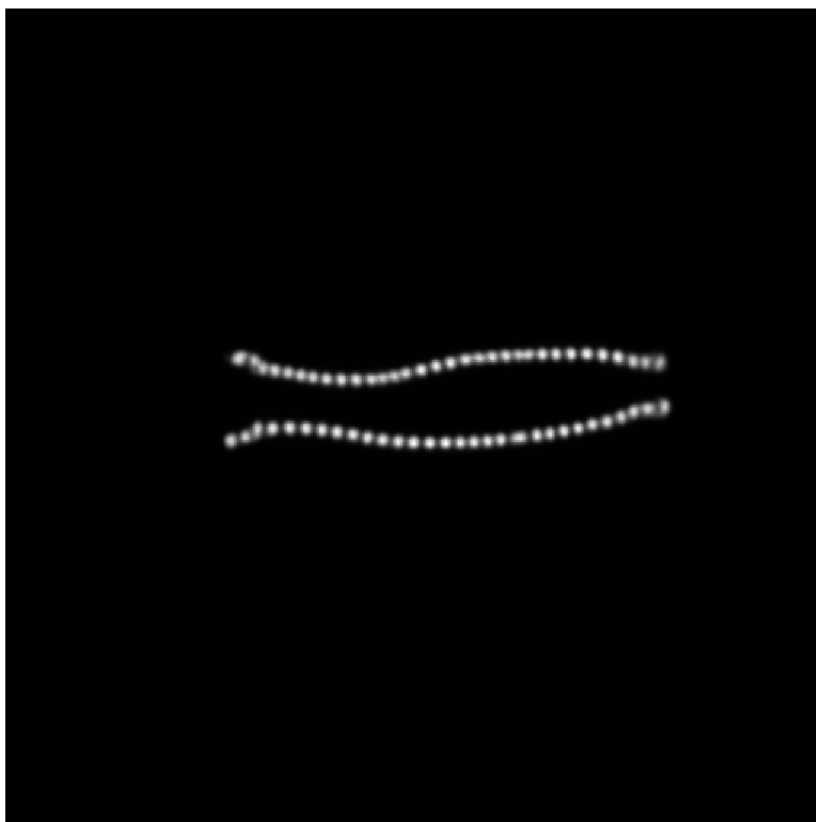
Rozměry vstupů je vhodné volit co největší, aby byly co nejviditelnější detaily v obrazu. Velké rozměry pak ale prodlužují čas trénování a odpovědi výsledného modelu. Rozměry  $512 \times 512$  byly zvoleny jako vhodný kompromis.

#### Generátor lokalizačních map

Architektura modifikované sítě U-Net byla iterativně měněna tak, aby její modely měly co nejlepší výsledky. Výsledná architektura na obr. 30 má pět úrovní, na vstupu přijímá barevný výřez  $Y$  o rozměrech  $512 \times 512 \times 3$ . Konvoluční vrstvy na první úrovni mají 16 konvolučních masek. Aby měly výstupní mapy a konvoluční vrstvy stejné rozměry, jsou konvoluční vrstvy doplněny o zero-padding. Modifikovaný U-Net vrací na výstupu lokalizační mapu sekundárních šupin  $\hat{\mathcal{L}}$  o rozměrech  $512 \times 512 \times 1$ . Příklad lokalizační mapy je na obr. 31.

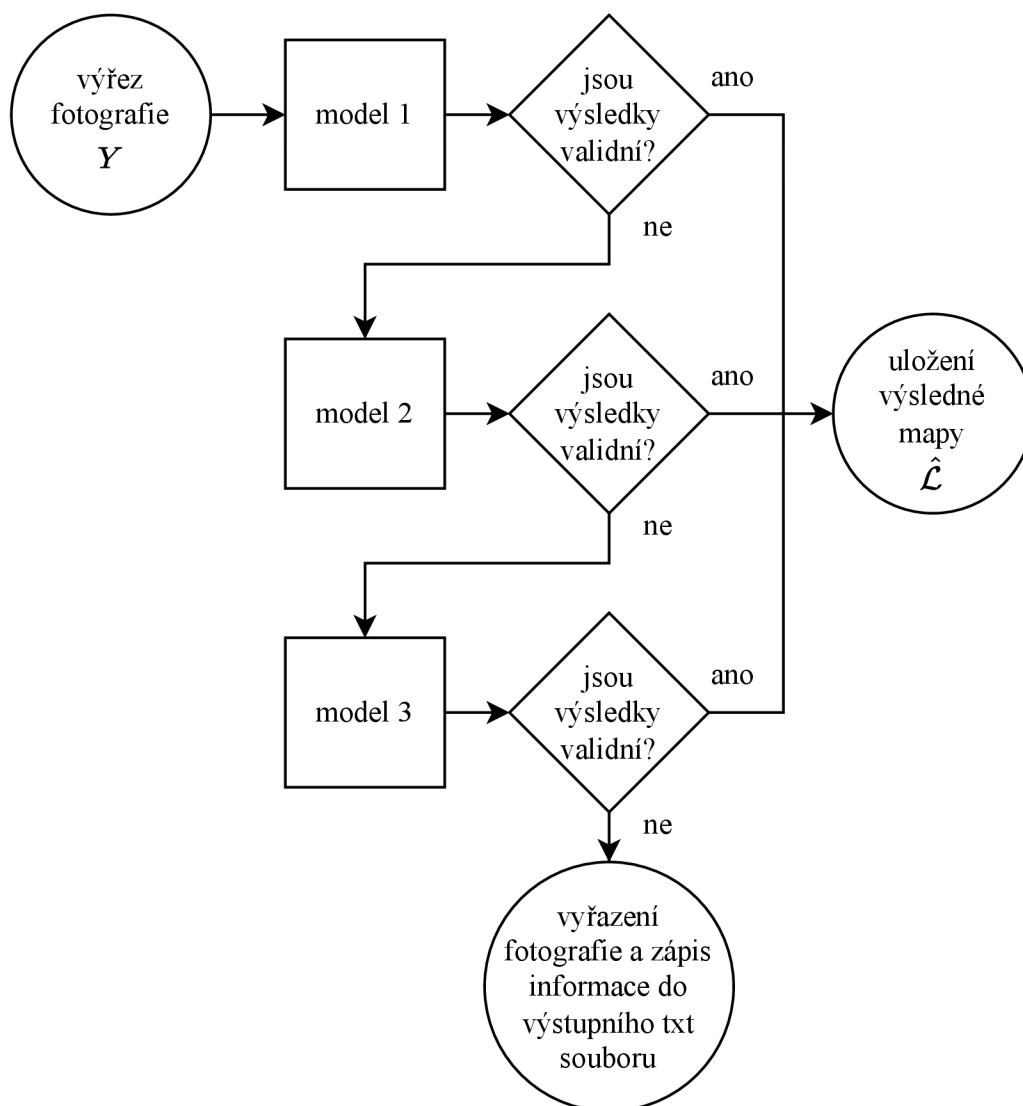


Obr. 30: Architektura modifikované sítě U-Net o pěti úrovních a 64 konvolučními maskami v konvolučních vrstvách první úrovně. Síť přijímá výřez  $Y$  o rozměrech  $512 \times 512 \times 3$  pixelů a vrací lokalizační mapu  $\hat{L}$  o rozměrech  $512 \times 512 \times 1$  pixelů s hodnotami 0 až 255.



Obr. 31: Příklad lokalizační mapy  $\hat{\mathcal{L}}$  o rozměrech  $512 \times 512$  výřezu  $Y$  z obr. 29 s hodnotami 0 (černá) až 255 (bílá). Lokální maxima jsou pozice detekovaných šupin.

Aby nebyla navyšována délka výpočtu, budou ve výsledném řešení vybrány nejvýše tři nejlepší natrénované modely sítě U-Net. Každý bude trénován s jinak upravenou datovou sadou (viz podkapitulu 4.4.2), a proto bude mít každý model jiný výstup pro stejnou vstupní fotografii  $I$ . Využití souborového učení (použití více modelů) zpřesní výsledky, protože bylo zjištěno, že každý jinak natrénovaný model pracuje lépe s jiným typem ještěrky. Pokud výstup prvního modelu nebude validní, systém použije jiný model. Validita výstupu se určuje maximálním a minimálním počtem detekovaných šupin, který uživatel programu nastaví vstupními parametry `MAXIMUM_SCALES` a `MINIMUM_SCALES` před spuštěním programu. Nenažde-li se v žádném z modelů validní řešení, systém zapíše informaci do výstupního txt souboru a zpracování fotografie je i pro další části systému strojového vidění ukončeno. Pořadí volání modelů bude určeno analýzou výsledků. Schéma použití tří nejlepších modelů je na obr. 32.

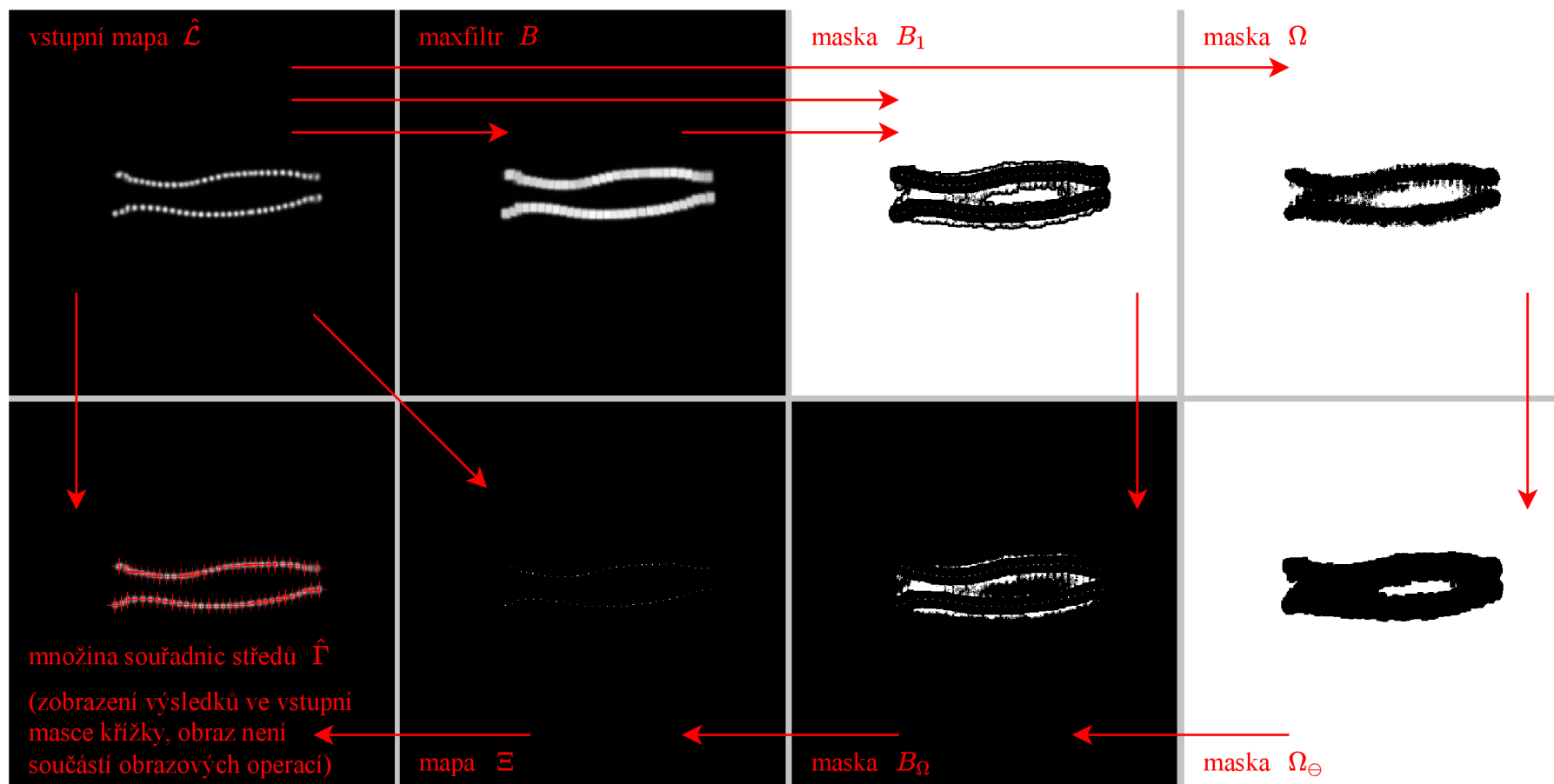


Obr. 32: Schéma použití souborového učení tří zvolených modelů.

### Centroid counterpoint

V lokalizační mapě  $\hat{\mathcal{L}}$  jsou pomocí Centroid counterpointu vyhledány souřadnice středů všech gradientů množiny  $\hat{\Gamma}$ . Uživatel programu zvolí hodnotu prahu  $t_m$  parametrem TM. Vizualizace metody pro lokalizační mapu je na obr. 33.

Uživatel programu zvolí pomocí parametru `MINIMUM_DISTANCE` minimální přípustnou vzdálenost v pixelech bodů z množiny  $\hat{\Gamma}$ . Z množiny jsou vymazány všechny body, které mají euklidovskou vzdálenost od ostatních bodů rovnu nebo menší zadanému parametru. Tím se zlepší výsledky u hůře vygenerovaných lokalizačních map gradientů v místech, kde je více lokálních extrémů na jedné šupině.



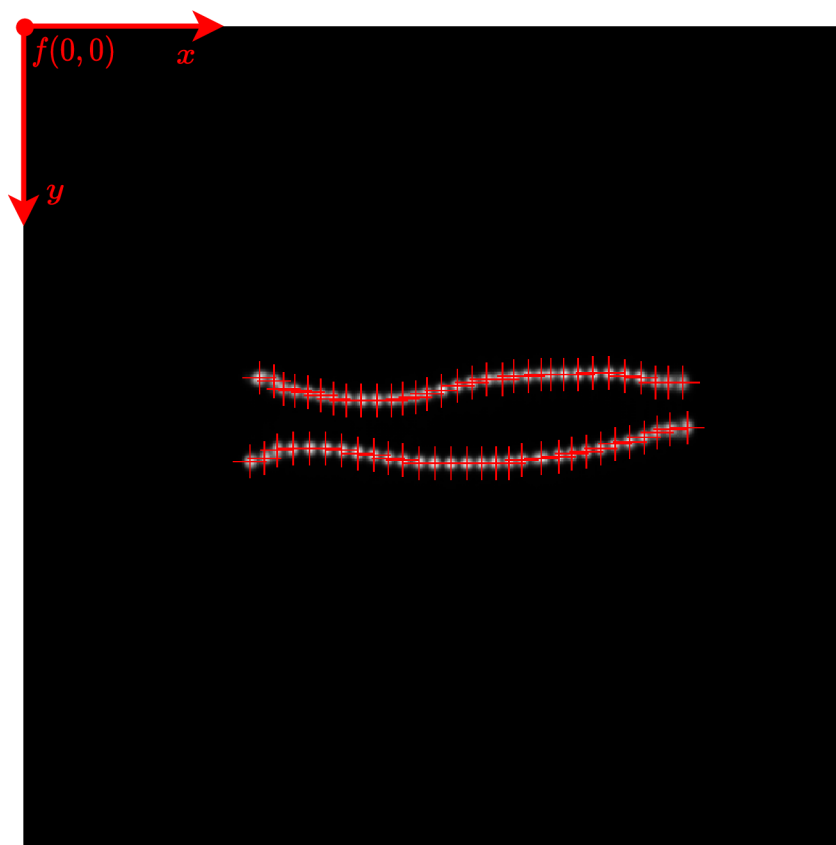
Obr. 33: Grafické znázornění Centroid counterpointu pro mapu z obr. 31. Rovnice pro zobrazené obrazové operace jsou popsány v podkapitole 3.8.2. Hodnota prahu  $t_m = 0,1$  a parametr MINIMUM\_DISTANCE byl zadán 2.

#### 4.2.4 Rozdělení detekovaných šupin na pravé a levé

Detekované souřadnice středů sekundárních šupin v množině  $\hat{\Gamma}$ , které jsou zobrazeny na obr. 34, je potřeba rozdělit na levé a pravé šupiny

$$\hat{\Gamma} \rightarrow \Psi_L, \Psi_P. \quad (44)$$

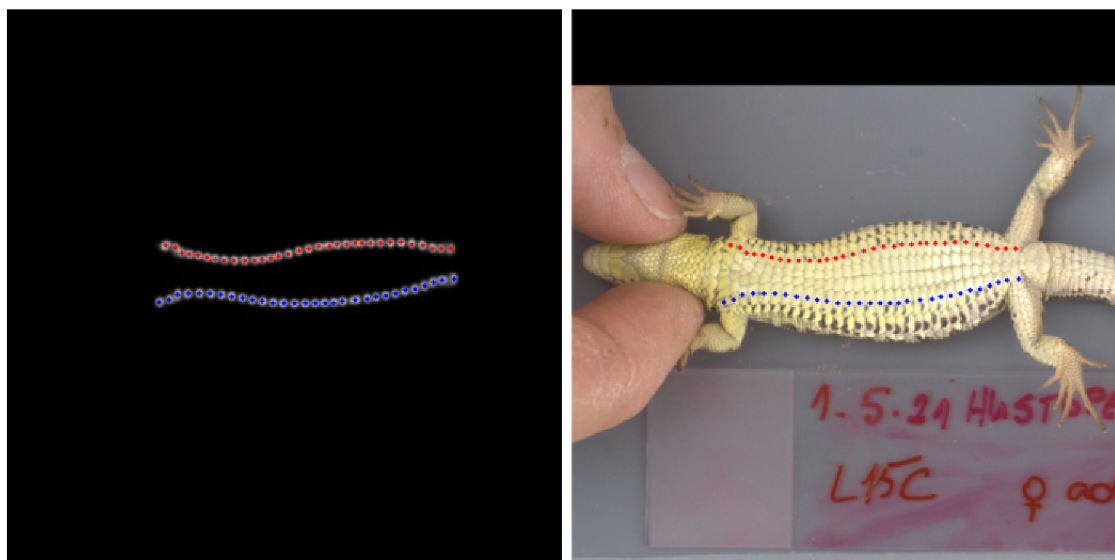
Jelikož jsou všechny ještěrky hlavou natočené na levou stranu, tak levá řada sekundárních šupin bude vždy spodní a pravá řada vždy vrchní pás bodů. Vzhledem k jednoduchosti problému bylo přistoupeno k prosté algoritmizaci namísto řešení pomocí neuronových sítí (například shlukovou analýzou).



Obr. 34: Ukázka detekovaných souřadnic středů šupin množiny  $\hat{\Gamma}$  v lokalizační mapě gradientů  $\hat{\mathcal{L}}$  z obr. 31. Červené křížky označují nalezené středy lokálních maxim gradientů. V levém horním rohu je znázorněn souřadný systém.

Navržená metoda vybere z množiny  $\hat{\Gamma}$  bod, jehož souřadnice  $x$  má ze všech vstupních bodů nejnižší hodnotu (tedy ten nejvíce nalevo). Následně najde jeho nejbližšího souseda euklidovskou metrikou. Iterativně hledá další nejbližší sousedy posledního nalezeného souseda. Vzdálenosti sousedů se ukládají a před hledáním dalšího souseda se počítá jejich aritmetický průměr  $p_{\emptyset}$ . Pokud je vzdálenost dalšího souseda alespoň dvakrát větší než  $p_{\emptyset}$ , hledání nejbližších sousedů se zastaví a znamená to, že další bod se už nachází v jiné řadě šupin. Experimentálně bylo zjištěno,

že právě dvojnásobek  $p_{\emptyset}$  je ideální prahová hodnota pro oddělení levé a pravé řady sekundárních šupin. Všechny doposud nalezené body se prohlásí za body řady jedné a zbylé body za body řady druhé. U obou řad je spočítána průměrná hodnota  $y$  souřadnice všech bodů. Řada s vyšší průměrnou hodnotou  $y$  všech svých bodů je následně označena jako levá (množina souřadnic  $\Psi_L$ ) a řada s nižší průměrnou hodnotou  $y$  jako pravá (množina souřadnic  $\Psi_P$ ). Výsledek je ukázán na obr. 35.



Obr. 35: Vizualizace rozdělení souřadnic středů detekovaných sekundárních šupin množiny  $\hat{\Gamma}$  na levou  $\Psi_L$  (modré tečky) a pravou  $\Psi_P$  (červené tečky) řadu. Nalevo jsou detekované šupiny zobrazeny v lokalizační mapě gradientů  $\hat{\mathcal{L}}$  (viz obr. 31) a napravo jsou zobrazeny ve výřezu  $Y$  (viz obr. 29). Fotografie na obrázku byla převzata a upravena z [4].

#### 4.2.5 Spočítání pravých a levých šupin

Vstupem do této části systému strojového vidění jsou množiny souřadnic levých a pravých řad detekovaných sekundárních šupin  $\Psi_L$  a  $\Psi_P$ . Prostým spočítáním délky polí, které množiny v programu reprezentují, je zjištěn počet

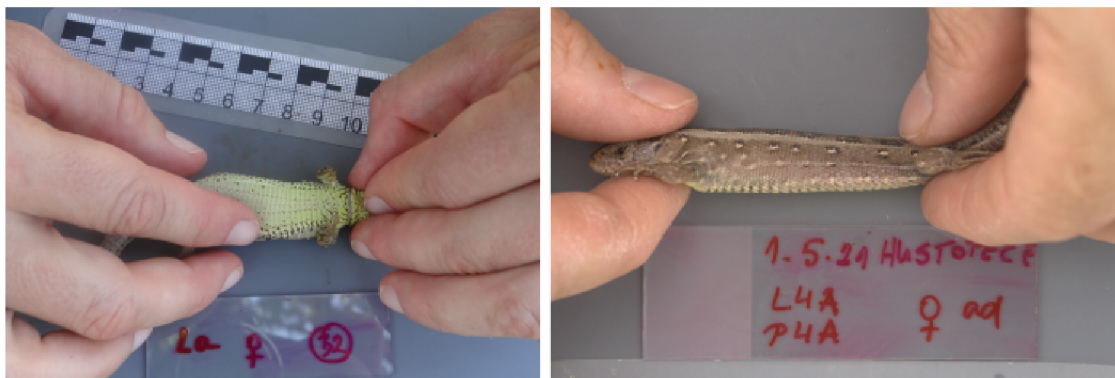
$$\Psi_L, \Psi_P \rightarrow |\Psi_L|, |\Psi_P|. \quad (45)$$

Systém strojového vidění tímto končí pro zadanou vstupní fotografii. Počty  $|\Psi_L|$  a  $|\Psi_P|$  jsou zapsány do výstupního txt souboru v požadovaném formátu, který je popsán v podkapitole 2.1.



### 4.3 Příprava datových sad

Ne všechny poskytnuté fotografie byly použitelné (viz obr. 36). Vyřazeny byly fotografie, na kterých ještěrky nebyly vyfocené z ventrální strany (asi 80 % z celkového počtu), a dále pak fotografie, na kterých nebyly viditelné všechny šupiny. Některé fotografie byly vyřazeny, protože byly duplicitně uloženy v různých formátech.



Obr. 36: Ukázka vyřazených fotografií [4]. Na levé fotografii nejsou viditelné všechny šupiny a na pravé fotografii není ještěrka vyfocená z ventrální strany.

K trénování modelů konvolučních neuronových sítí a vývoji ostatních částí systému strojového vidění popsaných v podkapitole 4.2 bylo vybráno 227 fotografií *I*, které nejsou součástí přílohy. Před trénováním bylo potřeba náhodně změnit jejich pořadí, fotografie chronologicky pojmenovat a sjednotit jejich formát. Pro tento úkol byl napsán skript `prepare.py`, který je v adresáři přílohy A. Na vstupu přijímá jakákoliv obrazová data ve formátech png, jpg, psd a CR2. Výstupem jsou obrazová data převedena na formát png, která jsou v náhodném pořadí přejmenována od 0.png do 226.png. Formát png byl vybrán proto, že je bezkompresní a umožňuje jednoduchou manipulaci ve volně dostupných nástrojích pro zpracování obrazu a strojového učení v jazyce Python 3.

Sada připravených fotografií byla rozdělena na trénovací sadu o 150 vzorcích a testovací sadu o 77 vzorcích. Rozdělení vzorků do sad se po celou dobu tvorby řešení neměnilo. Velikost trénovací sady odpovídá běžnému počtu vzorků pro trénování modelů konvolučních neuronových sítí. Testovací sada je dostatečně velká pro vyhodnocení výsledků.

#### 4.3.1 Štítkování hlavy, břicha a ocasu

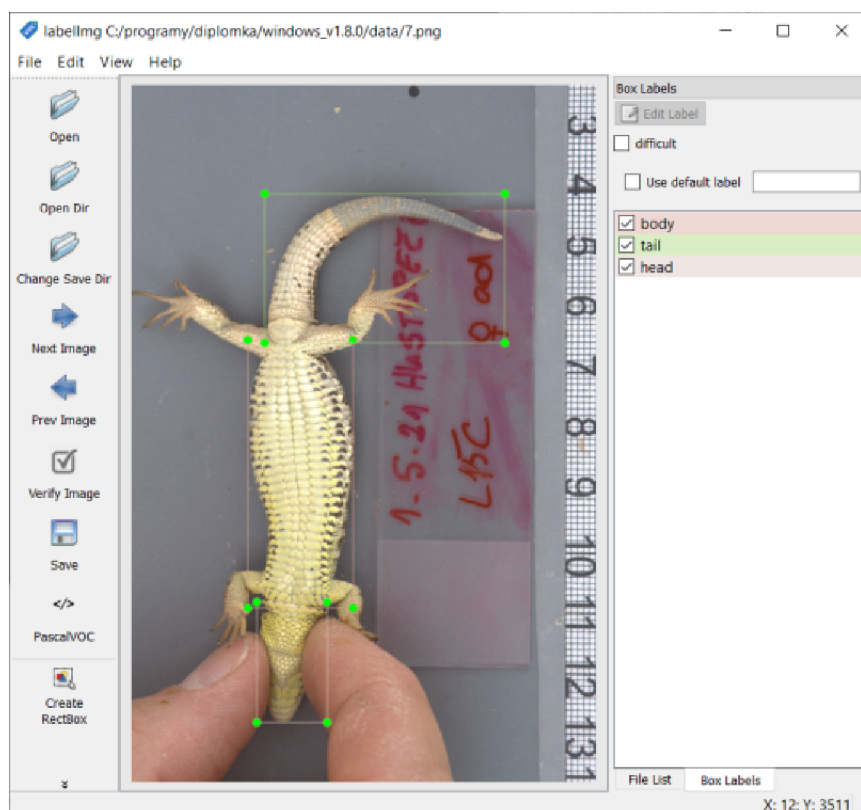
Štítkování trénovací sady 150 fotografií *I* pro detektor YOLOv4 probíhalo v programu LabelImg 1.8 pro Windows. Ukázka je na obr. 37. Testovací sada nemusela být pro zjištění přesnosti štítkovaná, protože se na ní hodnotila poloha objektů nutná k natočení a oříznutí fotografie a nikoliv hodnota metriky. Výstupem štítkovacího

programu LabelImg je stejnojmenný txt soubor pro každou fotografii, ve kterém jsou na řádcích uspořádané šestice očekávaných bounding-boxů

$$k = \left( n, x_c, \frac{x_{ul}}{M}, \frac{y_{ul}}{N}, \frac{x_{br}}{M}, \frac{y_{br}}{N} \right), \quad (46)$$

kde  $x_c = 1$  a  $k \in K$ .

Štítkovány byly třídy: body (břicho), tail (ocas) a head (hlava). Třídy byly číselně označeny takto:  $n = 0$  pro body,  $n = 1$  pro head a  $n = 2$  pro tail. Formát označování štítků je vyžadován trénovacím systémem darknet, který byl použitý pro trénování modelu YOLOv4.



Obr. 37: Štítkování břicha, ocasu a hlavy fotografie  $I$  v programu LabelImg. Fotografie na obrázku byla převzata a upravena z [4].

### 4.3.2 Štítkování šupin

Pro trénování modelů U-Net, které jsou součástí systému strojového vidění, bylo potřeba štítkovat souřadnice a velikosti gradientů šupin ve výřezech  $Y$ . Tyto souřadnice byly použity pro vykreslení očekávaných lokalizačních map  $\mathcal{L}$  vzorků jak u trénovací, tak i u testovací sady.

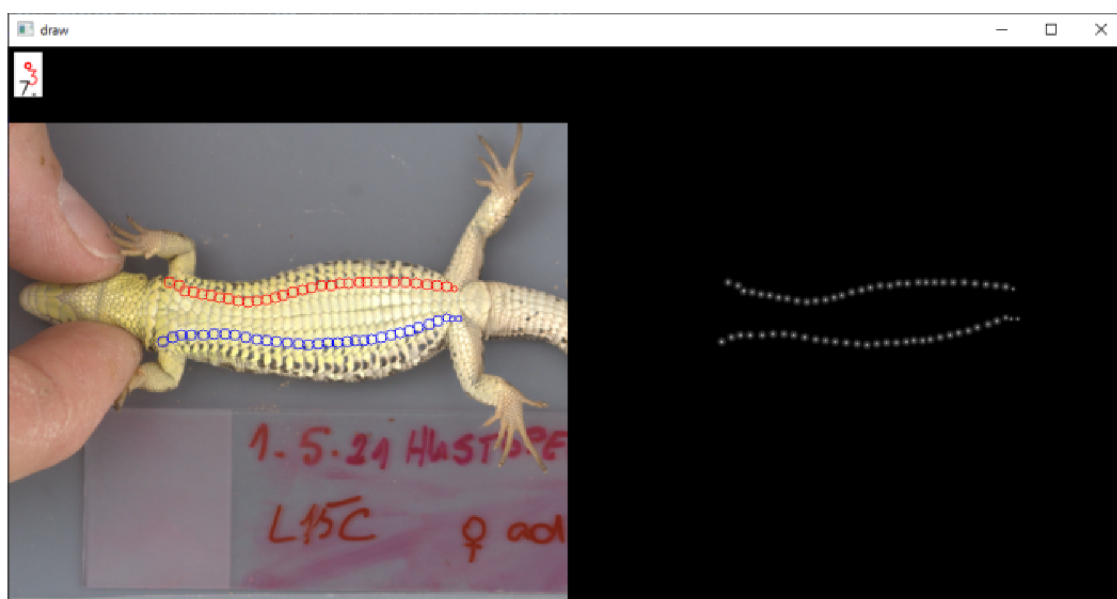
Jedná se o specifický úkol, pro který nebyl nalezen žádný volně dostupný štítkovací software. Proto musel být navržen štítkovací program. Vstupem do štítko-

vacího programu je sada orientovaných výřezů  $Y$  a výstupem je pickle soubor, který obsahuje informace o poloměrech gradientů  $r$  a pozicích

$$(n, x_{cd}, y_{cd}) \in \Gamma, \quad (47)$$

kde třída  $n$  je buď pravá nebo levá řada.

Po spuštění programu se načtou vstupní výřezy  $Y$ , mezi kterými se přepíná klávesami Q (zpět) a E (další). Levým kliknutím myši se štítkují šupiny, pravým kliknutím se maže poslední zadaná pozice a kolečkem se upravuje poloměr  $r$  gradientu podle velikosti šupiny. Data se ukládají průběžně. Ukázka programu je na obr. 38.



Obr. 38: Navržený štítkovací program. Nalevo je prostředí pro označování šupin myši (modrá kolečka jsou šupiny levé sekundární řady šupin, červená kolečka jsou šupiny pravé sekundární řady šupin). Napravo je vizualizace očekávané lokalizační mapy  $\mathcal{L}$  s hodnotami 0 (černá) až 255 (bílá). V levém horním rohu je barvou označená štítkovaná třída, první číslo zobrazuje poloměr gradientu v pixelech, druhé číslo pořadí výřezu  $Y$ .

Celkem bylo štítkováno všech 150 výřezů trénovací sady a 77 výřezů testovací sady. Výřezy byly získány spuštěním skriptu, který je dostupný v příloze A pod názvem `cut_body.py`. Štítkovací program je dostupný v příloze A pod názvem `labeling.py`. Soubor pickle se štítkami není součástí přílohy.

## 4.4 Implementace řešení

Podkapitola popisuje trénování modelů YOLOv4 a U-Net na oštitkované trénovací sadě.

### 4.4.1 Trénování modelu YOLOv4

Trénovací sada fotografií  $I$  byla rozdělena na trénovací podmnožinu 90 % vzorků a validační podmnožinu 10 % vzorků. Tento poměr byl experimentálně ověřen jako dostatečný. Byl použitý optimalizační algoritmus stochastic gradient decent (SGD) s momentovou větou. Hodnoty všech parametrů byly nastaveny tak, jak je doporučoval autor článku [47], ze kterého byl kód implementující trénovací systém darknet pro YOLOv4 převzat a upraven. K dosažení lepších výsledků byly hodnoty experimentálně měněny. Výčet nejdůležitějších parametrů použitých během trénování je v tab. 1.

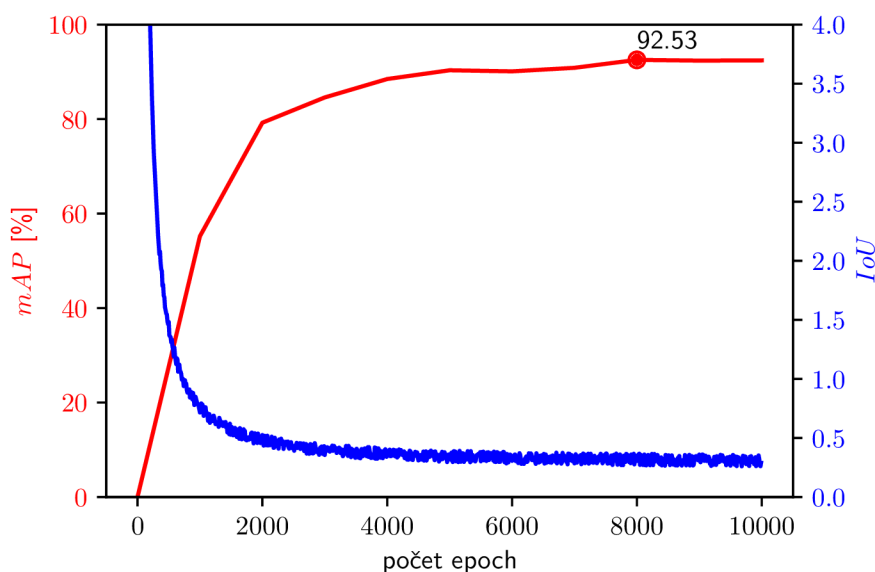
Tab. 1: Přehled nejdůležitějších vstupních parametrů pro trénování sítě YOLOv4.

parametr	proměnná v kódu	hodnota
velikost dávky [vzorků]	<code>batch</code>	64
rozměry vstupů $M \times N$ [pixel]	<code>width, height</code>	$416 \times 416$
learning rate $\mu$	<code>learning_rate</code>	0,001

Rozdělení vzorků do trénovací a evaluační podmnožiny se po každé epoše měnilo. Byla použita ztrátová funkce IoU (Intersection over Union). Hodnoty vah byly ukládány po 1 000 epochách a jen tehdy, pokud se zvýšila hodnota metriky mAP (mean Average Precision), která byla použita pro validaci. Vždy byl tak uložen jen doposud nejlepší natrénovaný model. Trénování výsledného modelu YOLOv4, který není součástí přílohy, nebylo dále ukládáno po 8 000 epochách. Graf vývoje ztrátové funkce a mAP hodnoty během trénování je na obr. 39.

Trénování modelu sítě YOLOv4 probíhalo v prostředí Google Colaboratory, což je online aplikace od společnosti Google, ve které se dají trénovat modely neuronových sítí na vzdálených počítačích s grafickými kartami Nvidia Tesla K80, T4, P4 a P100. Pracuje se soubory na Google Disku uživatele a Jupyter Notebooky, které uživatel edituje přímo v aplikaci. Služba je zdarma, trénování může být však přerušeno v jakémkoliv okamžiku. Proto bylo nutné implementovat kód, který zálohoval natrénované modely a v případě odpojení navázal na předchozí práci. Aplikace Google Colaboratory výrazně urychlila trénování všech modelů použitých v této práci. Na obr. 40 je struktura adresáře `yolov4/` s potřebnými konfiguračními soubory, který byl před trénováním vytvořen do kmenové složky Google Disku. Adresář `yolov4/` je v příloze A ve složce `google_colab/`.

Do složky `training/` se po ukončení trénování vygeneruje model ve formátu `weight`. V souborech `obj.data` a `obj.names` jsou nastaveny polohy souborů a názvy tříd. V souboru `obj.zip` jsou všechny fotografie trénovací sady a jejich štítkovací data ve formátu `txt` pospaná v podkapitole 4.3.1. V souboru `classes.txt` jsou na řádcích názvy štítkovaných tříd. Skript `process.py` náhodně rozdělí trénovací sadu na trénovací podmnožinu a validační podmnožinu. V konfiguračním souboru `yolov4-custom.cfg` je nastavena architektura sítě a parametry pro trénování. Trénování se spustilo příkazy v Jupyter Notebooku `yolov4_jupyter_train.ipynb` ve složce `google_colab/`, která je spolu s návodem `README.md` k obsluze v příloze A.



Obr. 39: Průběh trénování modelu YOLOv4. Červeně je průběh  $mAP$  (mean Average Precision) a modře hodnota ztrátové funkce  $IoU$  (Intersection over Union). Váhy se neukládaly po 8 000 iteracích, kde  $mAP = 92,53\%$  měla nejvyšší hodnotu.

```

yolov4/
|__training/
|__obj.data
|__obj.names
|__obj.zip
|   |__classes.txt
|__process.py
|__yolov4-custom.cfg

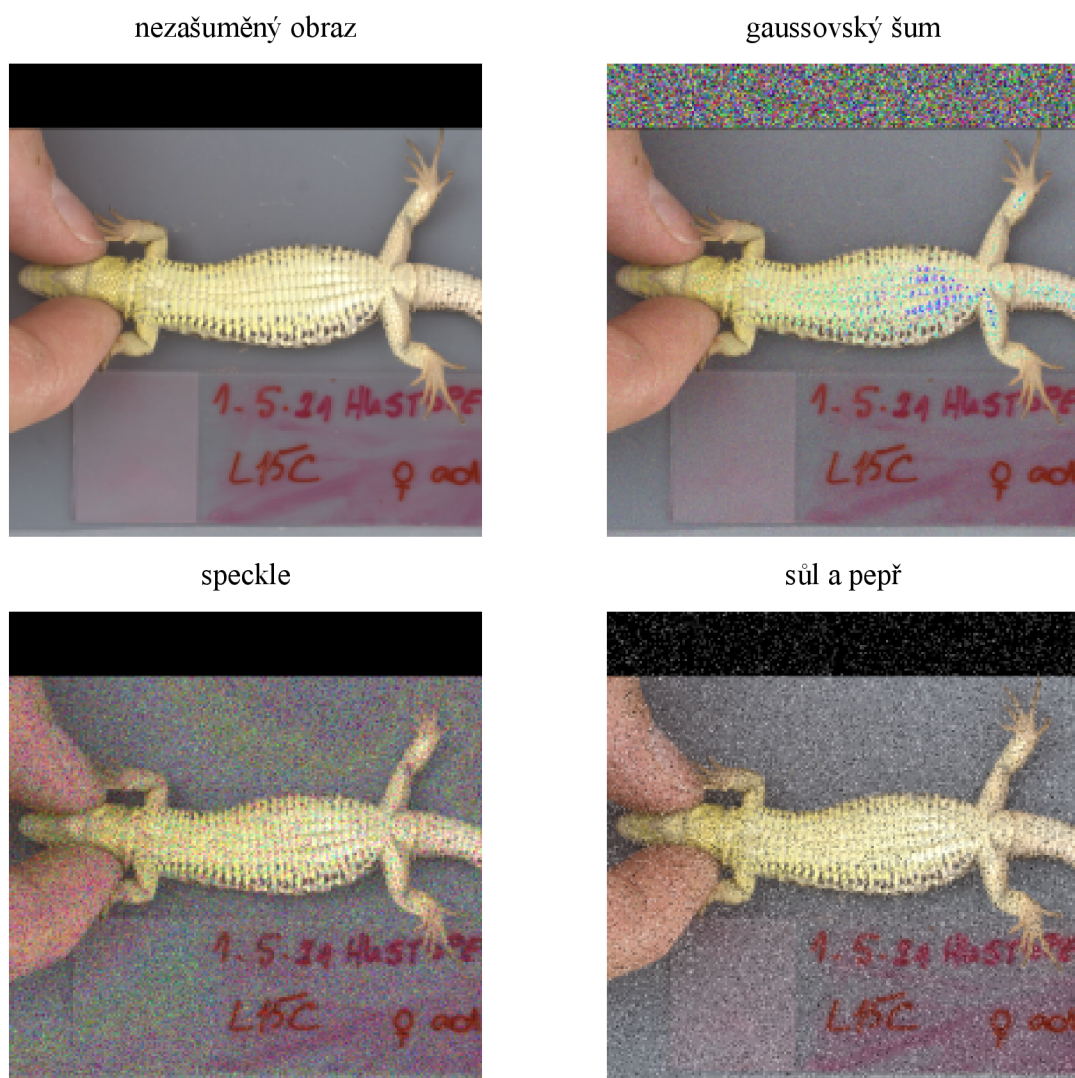
```

Obr. 40: Struktura adresáře `yolov4/`, která je nutná pro natrénování modelu YOLOv4 systémem `darknet`. Adresář je dostupný v příloze A ve složce `google_colab/`.

#### 4.4.2 Trénování modelů U-Net

Trénování probíhalo na trénovací sadě 150 oštitkovaných vzorků výřezů  $Y$ . Trénovací sada byla rozdělena na podmnožinu 85 % trénovacích vzorků a podmnožinu 15 % validačních vzorků.

Vstupní výřezy  $Y$  některých modelů byly augmentovány šumem. Mezi šumy patří: sůl a pepř (salt and pepper), gaussovský šum (gaussian noise) a speckle. Ukázka zašuměného výřezu včetně použitých paramterů je na obr. 41. Pro každý výřez byl při trénování náhodně vybírán jeden ze tří šumů nebo žádný se stejnou pravděpodobností 0,25. Zašuměn byl jen výřez  $Y$ , nikoliv očekávaná lokalizační mapa gradientů  $\mathcal{L}$ .



Obr. 41: Ukázka augmentace šumem obr. 29. Gaussovský šum s hodnotami  $q = 0$  a  $\sigma = 10$ . Speckle s hodnotami  $q = 0$  a  $\sigma = 0,7$ . Sůl a pepř s hodnotami  $P_s = 0,05$  a  $P_p = 0,05$ . Fotografie na obrázku byly převzaty a upraveny z [4].

Na výřezy  $Y$  některých modelů byly použity geometrické augmentace. Pro každý výřez byla náhodně vybrána jedna z geometrických augmentací v tab. 2 nebo žádná. Pravděpodobnost výběru byla pro všechny augmentace nebo žádnou augmentaci  $1/7$ . Hodnoty parametrů jednotlivých augmentací byly voleny náhodně v rozmezí hodnot v tab. 2. Augmentován byl jak výřez  $Y$ , tak očekávaná lokalizační mapa gradientů  $\mathcal{L}$ , a to vždy stejně.

Tab. 2: Popis vybraných modelů.

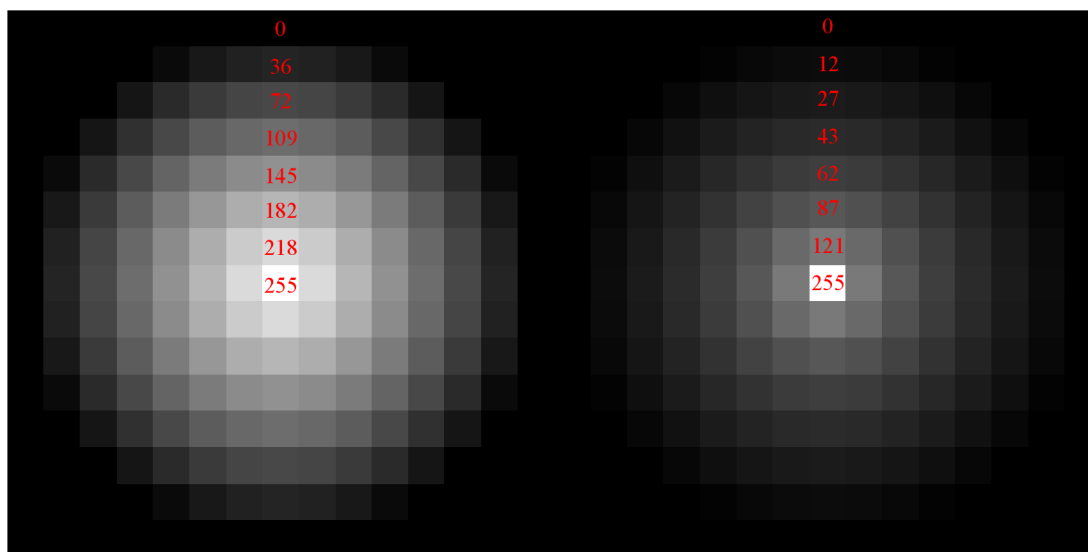
geometrická augmentace	rozmezí hodnot parametrů
rotace	$\theta \in (-0,028\pi; 0,028\pi)$ rad
translace ve směru osy x	$t_x \in (-M \cdot 0,1; M \cdot 0,1)$
translace ve směru osy y	$t_y \in (-N \cdot 0,1; N \cdot 0,1)$
vertikální převrácení obrazu	-
zkosení	$s_x, s_y \in (-0,15; 0,15)$
škálování	$c_x = c_y \wedge c_x \in (0,7;1)$

Pro zvýšení robustnosti systému byly některé modely trénovány s lineárně a jiné s nelineárně klesajícím gradientem v očekávané lokalizační mapě  $\mathcal{L}$  (pro ukázkou viz obr. 42). Hodnoty pixelů lineárně klesajícího gradientu byly vypočítány rovnicí

$$m(d) = -\left(\frac{255}{r}\right) \cdot d + 255, \quad (48)$$

kde  $m(\cdot)$  je funkce závislosti hodnoty pixelu vzdálenosti  $d$  pixelu od středu gradientu a  $r$  je poloměr gradientu. Hodnoty pixelů nelineárně klesajících gradientů byly dány vztahem

$$m(d) = -255 \cdot \left(\frac{d}{r}\right)^{\frac{1}{3}} + 255. \quad (49)$$



Obr. 42: Vlevo je ukázka lineárně klesajícího gradientu popsáno rovnicí (48) a vpravo je nelineárně klesající gradient popsaný rovnicí (49). Červeně jsou popsány hodnoty vybraných pixelů. Oba gradienty mají poloměr  $r = 7$  pixelů.

Aby bylo nalezeno co nejvíce vhodných modelů, byly během trénování zkoušeny různé konfigurace, které se odlišovaly:

- počtem úrovní sítě U-Net,
- počtem konvolučních masek v konvolučních vrstvách na první úrovni sítě U-Net,
- přítomností augmentace náhodně vybraným šumem,
- přítomností náhodně vybranou geometrickou transformací,
- volbou mezi lineárním a nelineárním gradientem,
- fixním poloměrem gradientu o hodnotě  $r = 5$  a poloměrem nastaveným při štítkování očekávaných lokalizačních map  $\mathcal{L}$ ,
- trénováním U-Net sítě pro dvě třídy  $n$  (pravé a levé šupiny) a jednu třídu  $n$  (všechny šupiny),
- hodnotou learning ratu  $\mu$
- a volbou ztrátové funkce.

Modely byly trénovány v aplikaci Google Colaboratory stejně jako YOLOv4. Implementace trénovacího algoritmu byla inspirována tvorbou doktora Bhattiprolu [48]. Hodnoty vah modelů byly ukládány jen po epochách, po kterých měly doposud nejlepší hodnotu metriky (ta byla vždy totožná s vybranou ztrátovou funkcí).

Do kmenové složky Google Disku byl vytvořen adresář `unet/` s potřebnými soubory, jehož struktura je na obr. 43. Je dostupný ve složce `google_colab/` v příloze A. Do složek `train_data_images/` a `test_data_images/` byly umístěny výřezy  $Y$  trénovací a testovací sady. Soubor `images_data.pickle` obsahuje štítkovací in-



formace vytvořené navrženým programem pro štítkování šupin, který je pospaný v podkapitole 4.3.2. Soubor pickle není součástí přílohy. Natrénované modely se ukládaly do složky `models/`. Konfigurace trénovaných modelů se nastavují v souboru `unet_jupyter_train.ipynb`, který je v příloze A ve složce `google_colab/`. Spuštěním příkazů v souboru `unet_jupyter_train.ipynb` podle přiloženého návodu `README.md` se spouští trénování.

```
unet/  
|__models/  
|__test_data_images/  
|__train_data_images/  
|__images_data.pickle
```

Obr. 43: Struktura adresáře `unet/`, která je nutná pro natrénování modelů U-Net. Adresář je umístěný ve kmenové složce Google Disku. Soubor `images_data.pickle` je na obrázku jen pro ilustraci a není součástí přílohy A.



## 5 VÝSLEDKY A JEJICH VYHODNOCENÍ

Tato kapitola popisuje implementaci systému strojového vidění do programu včetně výběru natrénovaných modelů U-Net.

### 5.1 Výběr modelů U-Net

Celkem bylo natrénováno 34 modelů, které nejsou součástí přílohy. Ty, jejichž výstupy  $\hat{\mathcal{L}}$  se vůbec nepodobaly očekávaným výstupům  $\mathcal{L}$ , byly ihned vyřazovány. Výběr vhodných modelů pro systém strojového vidění probíhal pozorováním výsledků modelů různých konfigurací na trénovací sadě 77 vzorků.

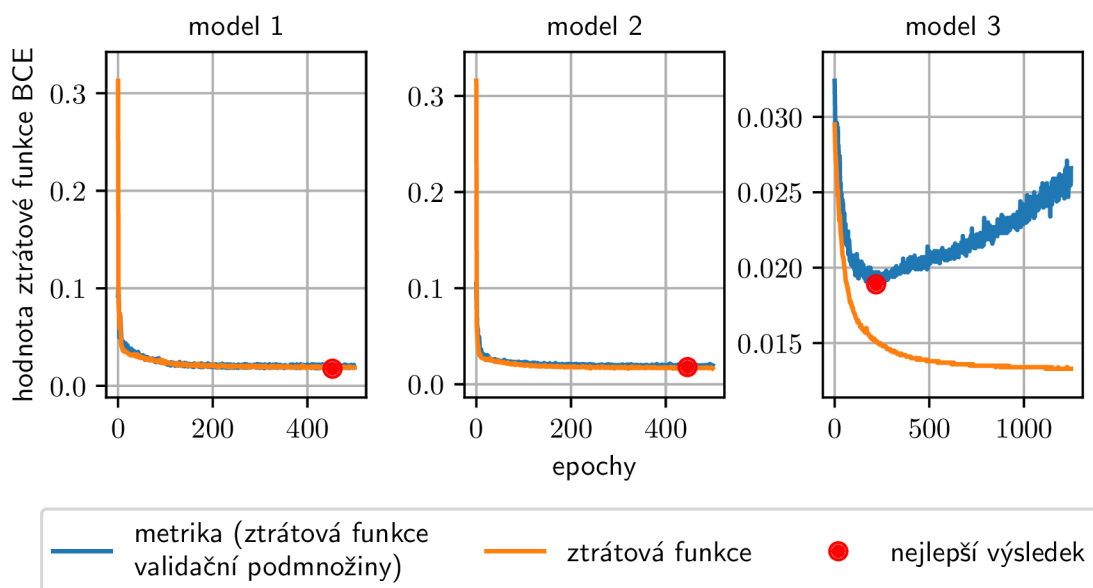
Počet úrovní sítě U-Net a počet konvolučních masek v konvolučních vrstvách na první úrovni ovlivňují jak přesnost modelu, tak i jeho rychlost, protože roste celkový počet trénovacích parametrů. U-Net sítě se čtyřmi úrovněmi neměly dobré výsledky, ačkoli byly rychlé. Výsledky sítí o pěti a šesti úrovních se od sebe téměř neodlišovaly, sítě s pěti úrovněmi byly však mnohem rychlejší. Proto byly vybírány jen modely o pěti úrovních. Nejlepší výsledky s ohledem na přesnost a rychlost měly sítě s 16 konvolučními maskami v jedné konvoluční vrstvě na první úrovni. Modely trénované pro dvě třídy  $n$  měly vůbec nejhorší výsledky, a proto byly vyřazeny. Learning rate  $\mu$ , který vedl k nejlepším výsledkům, měl hodnotu  $\mu = 0,001$ , a proto byly ponechány jen modely natrénované s touto hodnotou. BCE se ukázala jako jediná použitelná ztrátová funkce. Celkem bylo vybráno 5 vhodných modelů, v jejichž výstupech  $\hat{\mathcal{L}}$  bylo možné vyhledávat polohy gradientů, a zbylé byly vyřazeny.

Počty levých a pravých šupin  $|\Psi_L|$  a  $|\Psi_P|$  detekovaných na trénovací sadě vhodnými modely byly porovnány s očekávanými hodnotami. Poté byla analyzována přesnost jednotlivých modelů (tedy rozdíl detekovaných a očekávaných počtů šupin). Analýzou bylo zjištěno, že každý zkoumaný model je přesnější pro jiný typ ještěrky. Například modely trénované na  $\hat{\mathcal{L}}$  s fixním poloměrem gradientu  $r = 5$  byly přesnější u vzorků ještěrek s většími šupinami a modely trénované s nastavenými poloměry gradientů  $r$  během štítkování byly zase přesnější u vzorků ještěrek s malými šupinami.

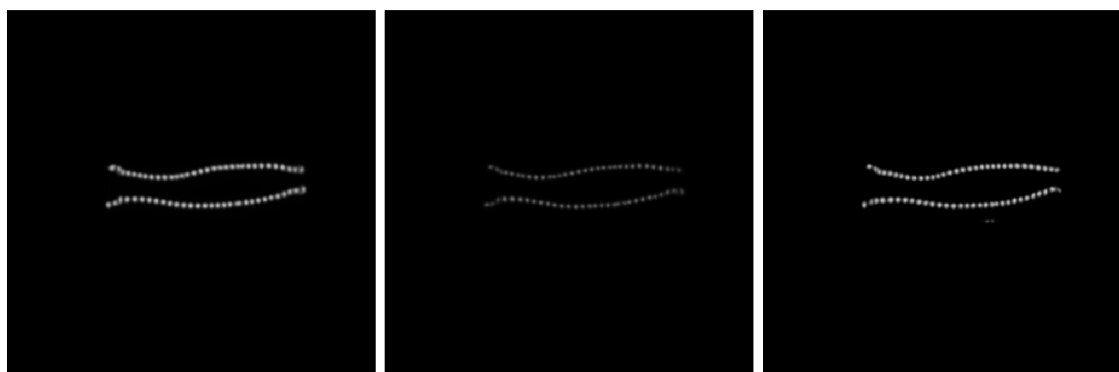
Na základě zjištění analýzy přesnosti vhodných modelů byly vybrány 3 modely tak, aby jejich kombinace metodou souborového učení, která je vizualizovaná na obr. 32 a popsána v podkapitole 4.2.3, dávala co nejpřesnější výstupní počty  $|\Psi_L|$  a  $|\Psi_P|$ . Konfigurace parametrů nastavených během trénování vybraných modelů jsou popsány v tab. 3. Hodnoty ztrátových funkcí a metrik během trénování jsou na obr. 44 a ukázka jejich výstupů  $\hat{\mathcal{L}}$  je na obr. 45.

Tab. 3: Parametry nastavené během trénování vybraných modelů.

	model 1	model 2	model 3
poloměr gradientu $r$ [px]	$r = 10$	$r \in (3; 7)$	$r \in (3; 7)$
charakter gradientu $m(\cdot)$	lineární	nelineární	lineární
náhodné augmentace šumem	ne	ano	ne
náhodné geometrické augmentace	ano	ano	ne



Obr. 44: Průběhy trénování vybraných modelů 1, 2 a 3.

Obr. 45: Výstupní lokalizační mapy  $\hat{\mathcal{L}}$  vybraných modelů 1, 2 a 3 (zleva). Vstupní výřez  $Y$  je na obr. 29.

## 5.2 Výsledný program

Program na vstupu přijme adresář s fotkami ještěrek. V adresáři mohou být fotky roztrženy do dalších adresářů. Program načte všechny fotografie ve formátech jpg, psd a CR2. Každá fotografie  $I$  je systémem strojového vidění, který je popsán v podkapitole 4.2, převedena na odhad počtu levých a pravých sekundárních šupin  $|\Psi_L|$  a  $|\Psi_P|$ . Tyto hodnoty program uloží do výstupního txt souboru podle požadavků v podkapitole 2.1. Pokud uživatel zvolí možnost vizualizace výsledků, tak do výstupního adresáře program uloží výřezy  $Y$  s barevně vyznačenými pozicemi detekovaných šupin ve formátu png tak, jak jsou jim odpovídající vstupní fotografie  $I$  řazeny ve vstupním adresáři. Vizualizace odpovídá obr. 35 vpravo.

Program byl napsán v jazyce Python 3.8.10. Chod programu lze před jeho spuštěním nastavit těmito vstupními parametry:

- CONFIDENCE\_THRESHOLD: paramter  $t_{bb}$  metody non-maxima suppression,
- ENLARGE\_BODY: poměr velikosti vyřezávaného obrazu a velikosti detekovaného břicha,
- CUT\_SIZE: rozměry  $M \times N$  vstupního výřezu  $Y$  do sítě U-Net, kde  $M = N$ ,
- LOCAL\_FILTER: rozměry čtvercového maxfiltru  $M_B \times N_B$ , kde  $M_B = N_B$ ,
- MINIMUM\_DISTANCE: minimální přípustná vzdálenost bodů z množiny  $\hat{\Gamma}$ ,
- TM: práh Centroid counterpointu  $t_m$ ,
- MAXIMUM\_SCALES: horní mez počtu šupin ještěrky, pro který program vyhodnotí detekci jako validní,
- a MINIMUM\_SCALES: spodní mez počtu šupin ještěrky, pro který program vyhodnotí detekci jako validní.

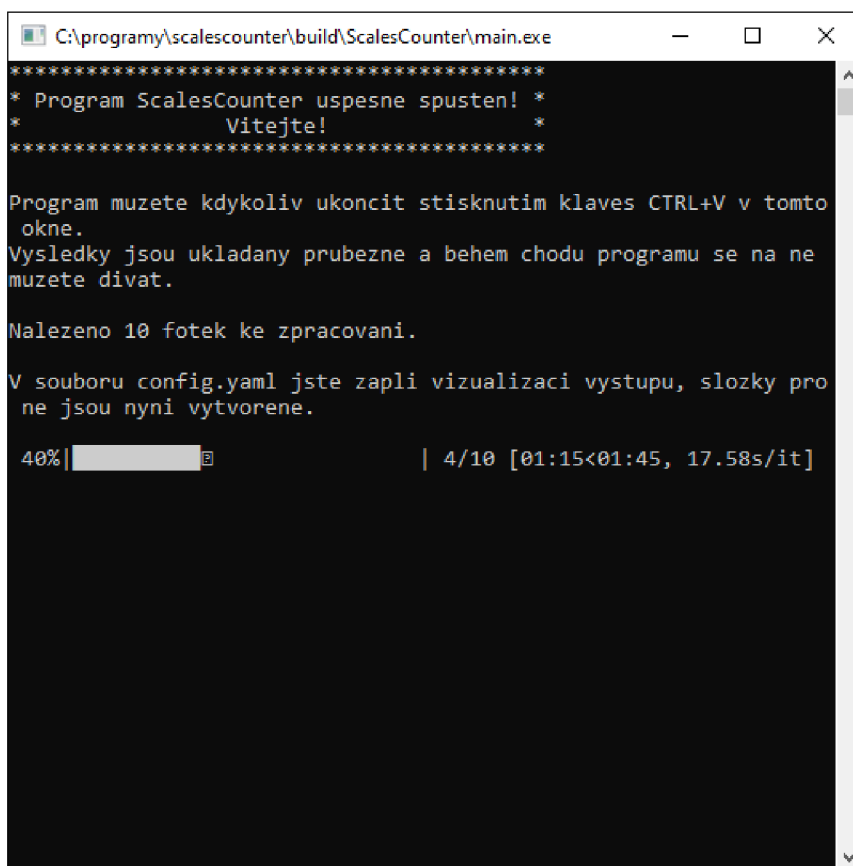
Zbylé proměnné, které byly v této práci zmíněny, jsou v programu nastaveny na určitou hodnotu, protože by jejich změna nevedla k lepším výsledkům. Jde o:

- horizontální rozměr okolí eroze  $M_\epsilon = 3$
- a vertikální rozměr okolí eroze  $N_\epsilon = 3$ .

Byly vytvořeny dvě verze programu, jedna pro lokální spuštění na operačních systémech Windows a druhá pro cloudové spuštění z webového prohlížeče v aplikaci Google Colaboratory. Obě verze programu fungují stejně, implementují stejný systém strojového vidění, obsahují stejné modely, jen se odlišně spouštějí.

### 5.2.1 Lokální verze pro Windows

Soubory potřebné pro sestavení lokální verze programu jsou v příloze B ve složce local/. Výsledná aplikace ve formátu exe zobrazená na obr. 46 se ovládá pomocí konfiguračního souboru config.yaml. Před spuštěním aplikace je nutné vedle exe souboru umístit i adresář se vstupy. Podrobný návod pro sestavení a spuštění je v souboru README.md ve složce local/ přílohy B.



```
C:\programy\scalescounter\build\ScalesCounter\main.exe
*****
* Program ScalesCounter uspesne spusten! *
*           Vitejte!           *
*****

Program muzete kdykoliv ukoncit stisknutim klaves CTRL+V v tomto
okne.
Vysledky jsou ukladany prubezne a behem chodu programu se na ne
muzete divat.

Nalezeno 10 fotek ke zpracovani.

V souboru config.yaml jste zapli vizualizaci vystupu, slozky pro
ne jsou nyní vytvorene.

40%|███████████| 4/10 [01:15<01:45, 17.58s/it]
```

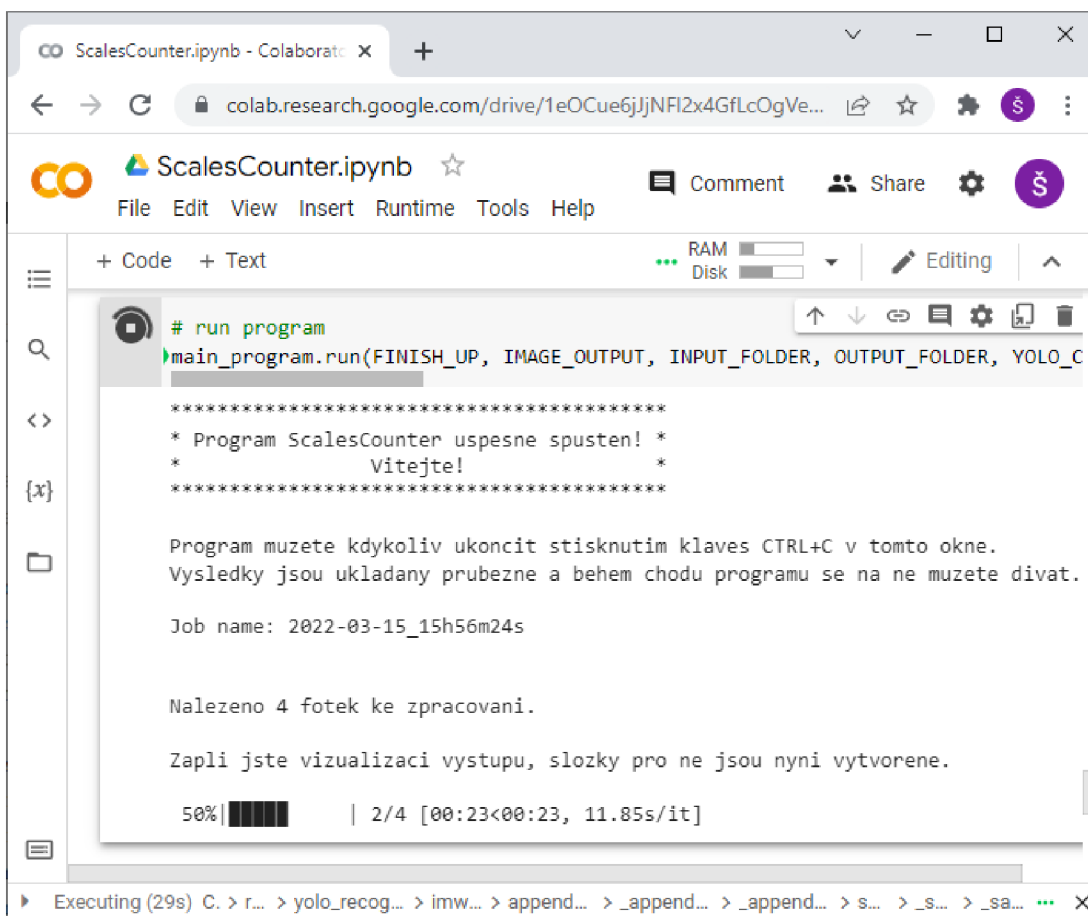
Obr. 46: Ukázka běhu lokální verze programu pro Windows.

### 5.2.2 Cloudová verze

Cloudová verze programu se spouští příkazy v Jupyter Notebooku přímo v aplikaci Google Colaboratory. Jupyter Notebook `ScalesCounter.ipynb` je k dispozici v příloze B ve složce `cloud/`. Obsah adresáře `cloud/ScalesCounter/` v příloze B se nakopíruje do Google Disku stejného účtu, pod kterým je aplikace Google Colaboratory spouštěna. Pomocí instrukcí v příloženém souboru `README.md` ve složce `cloud/` přílohy B se pak program nakonfiguruje. Spuštěním všech oken v souboru `ScalesCounter.ipynb` se program spustí. Výsledky se uloží na Google Disk.

## 5.3 Vyhodnocení

Po implementaci navrženého řešení se program otestoval na jeho přesnost a rychlost. Vyhodnocení výsledků proběhlo na testovací sadě 77 fotografií ještěrek v různých vstupních formátech pomocí skriptu `thesis_evaluation.py` v příloze A. Nastavení vstupních parametrů programu během experimentu je v tab. 4.



Obr. 47: Ukázka běhu cloudové verze programu.

Tab. 4: Hodnoty parametrů nastavené během experimentu.

parametr	hodnota
CONFIDENCE_THRESHOLD	0,18
ENLARGE_BODY	1,42
CUT_SIZE	512
LOCAL_FILTER	7
MINIMUM_DISTANCE	2
TM	0,1
MAXIMUM_SCALES	35
MINIMUM_SCALES	26

### 5.3.1 Přesnost řešení

Maximální přípustná průměrná odchylka detekovaného počtu šupin od skutečného počtu byla v požadavcích v podkapitole 2.1 stanovena na 2 šupiny na jednu fotografii ještěrky. Relativní a absolutní četnosti odchylek vzniklých při zpracování fotografií testovací sady programem jsou v tab. 5, jejich statistické zhodnocení pak v tab. 6.

Zhodnocení výsledků v tab. 6 ukazuje, že požadovaná přesnost programu byla splněna. Aritmetický průměr odchylek detekovaných a skutečných počtů šupin na jednu ještěrku je dokonce nižší, než bylo požadováno. Nejčastější odchylka měla hodnotu 1. Rozptyl hodnot není velký, což plyne i z tab. 5. Jedna fotografie byla vyřazena, protože u ní ani jeden U-Net model nedetekoval validní počet šupin.

Tab. 5: Četnosti odchylek detekovaného počtu šupin od skutečného počtu na testovací sadě 77 fotografií.

rozdíl detekovaného počtu šupin od skutečného	absolutní četnost	relativní četnost
0	12	16,22 %
1	32	43,24 %
2	17	22,97 %
3	8	10,81 %
4	2	2,70 %
5	1	1,35 %
6	1	1,35 %
chyba	1	1,35 %

Tab. 6: Statistické zhodnocení odchylek detekovaného počtu šupin od skutečného počtu na testovací sadě 77 fotografií.

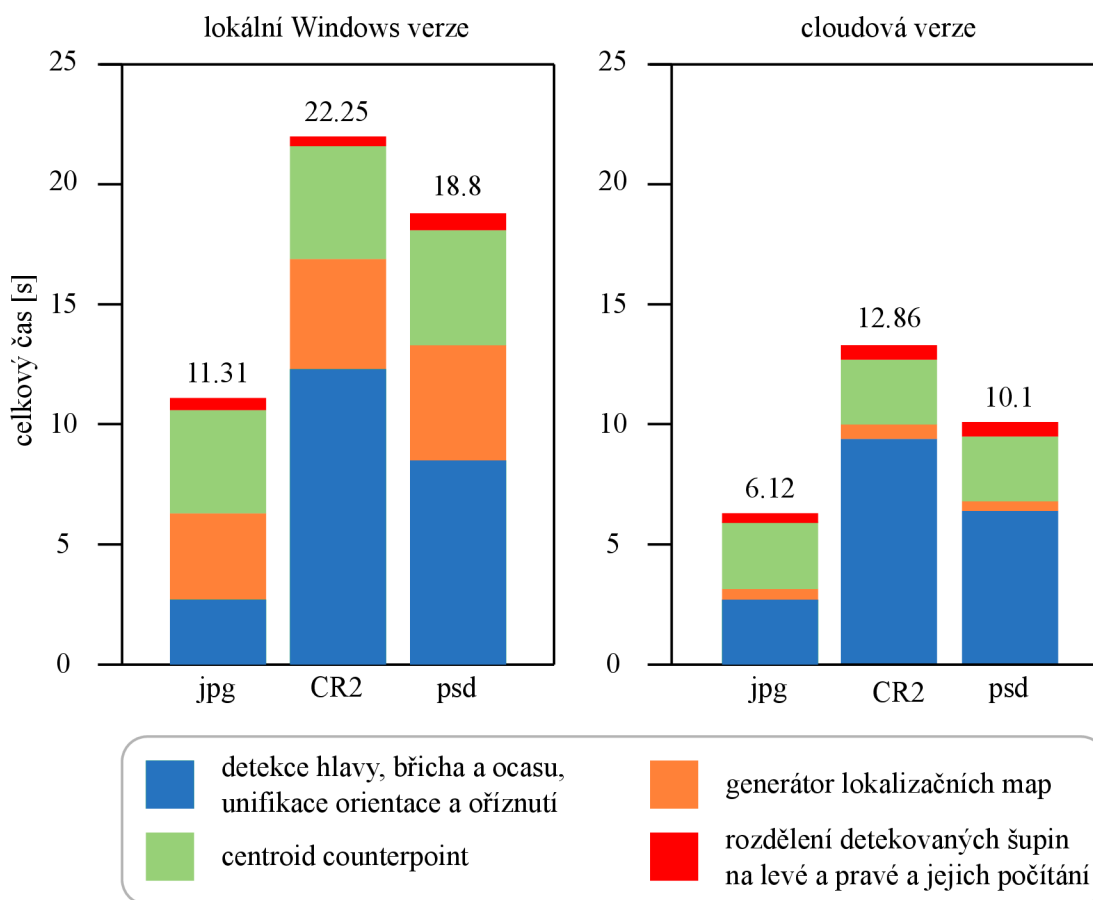
veličina	hodnota
aritmetický průměr	1,51
medián	1,00
rozptyl	1,40

### 5.3.2 Rychlost výpočtu

Rychlost zpracování jedné fotografie je důležitou informací zejména pro uživatele. Mimo hrubý odhad času trvání na posuvném ukazateli ve spuštěném programu na obr. 46 a 47 byla provedena podrobnější analýza časové náročnosti jednotlivých částí



systemu strojového vidění na všech 77 testovacích fotografiích. V grafu na obr. 48 je porovnání časů zpracování jedné fotografie jak lokální tak cloudové verze programu.



Obr. 48: Časová analýza průměrné délky zpracování jedné fotografie všech vstupních formátů jak lokální (vlevo), tak cloudové (vpravo) verze programu. Celkový čas výpočtu je barevně rozdělen do částí systému strojového vidění.

Z grafu na obr. 48 plyne, že cloudová verze je téměř dvakrát rychlejší pro všechny zkoumané formáty fotografií než verze lokální. Rychlost lokální verze je závislá na zařízení, na kterém program běží. Zkouška rychlosti lokální verze programu probíhala na počítači HP Pavilion x360 14-cd0013nc s procesorem Intel Pentium 4415U 2.30 GHz a 8 GB paměti RAM. Zkouška rychlosti cloudové verze programu proběhla na vzdálených serverech společnosti Google.

Jediná část systému strojového vidění, u které se výrazně liší poměrná délka výpočtu vůči ostatním částem, je detekce hlavy, břicha a ocasu, unifikace orientace a ořiznutí (viz obr. 48 modře). To je způsobeno tím, že tato část na začátku přijímá vstupní fotografie v různých formátech. Časově nejvýhodnější je zpracovávat fotografie ve formátu jpg. Ostatní části systému strojového vidění mají pro různé formáty vstupních fotografií jen minimální rozdíl v délce trvání.



## 6 DISKUSE

Výsledky navrženého programu naplnily všechny požadavky vědců z ÚBO AV ČR v.v.i. a PdF MUNI zejména pak požadavek na průměrnou odchylku detekovaného počtu sekundárních šupin od skutečného počtu. Nejvyšší průměrnou odchylku stanovili vědci na 2 šupiny na jednu fotografii ještěrky (viz podkapitulu 2.1), dosažená průměrná odchylka měřená na sadě 77 testovacích fotografií je 1,51.

Unifikací orientace a vyřiznutím zájmové oblasti ze vstupních fotografií pro generátor lokalizačních map, který je založený na architektuře sítě U-Net, se zvýšila přesnost detekce šupin. Centroid detektor generuje na pozice sekundárních šupin klesající gradienty s maximem uprostřed. Díky tomu je program schopný určit pozice šupin, které jsou velmi blízko u sebe. Přesnost programu byla výrazně zvýšena i implementací souborového učení. Přesnost by šla zvýšit rozšířením datové sady o nové oštitkované fotografie. Analýzou výsledků bylo zjištěno, že rozdělování detekovaných šupin na levou a pravou řadu bylo nejčastější příčinou vysokých odchylek, zejména pokud byly body na koncích řad příliš blízko u sebe. Mimo prosté algoritmizace existují i metody rozdělení shluků bodů pomocí strojového učení. Tyto metody by mohly být v budoucnu vyzkoušeny a porovnány se současným řešením. Výsledky by mohly být zpřesněny například i paralelním používáním obou metod a využít tak souborové učení i v této části systému strojového vidění.

Cloudová verze zpracuje jednu fotografii za 6,12 sekund (formát jpg), 10,1 sekund (formát psd) a 12,86 sekund (formát CR2). Člověk v rychlosti počítání šupin nemůže konkurovat navrženému programu. Systém strojového vidění byl implementován v jazyce Python 3, který slouží zejména k vývoji a testování softwaru. Pokud by byl systém strojového vidění do programu implementovaný jazykem nižší úrovně, jako je například C++, výpočetní časy by se pravděpodobně zkrátily.

Způsob, jakým se program ovládá, odpovídá požadavkům. Uživatelsky více přívětivé prostředí nebylo cílem této práce. Pokud by se podařilo rychlost vyhodnocování fotografií výrazně zvýšit, tak by program mohl být použitý v reálném čase přímo v terénu při odchyťování ještěrek. To by vyžadovalo implementovat program i na operační systém pro mobilní zařízení jako je například Android.



## 7 ZÁVĚR

Cílem diplomové práce bylo navrhnout a implementovat funkční program pro počítání sekundárních šupin v obrazových datech ventrálních stran těl ještěrek obecných. Navržený a implementovaný program je funkční a detekuje počty šupin s průměrnou odchylkou 1,51 od skutečného počtu. Požadavek byl nepřesáhnout odchylku 2 šupin. Všechny ostatní požadavky na podobu programu vymezené v kapitole 2 byly také naplněny. Cíl této diplomové práce byl tedy úspěšně splněn.

Výsledky práce zcela naplnily požadavky vědců z ÚBO AV ČR v.v.i. a PdF MUNI. Výsledný program vykazuje dokonce lepší výsledky. Vědci v rámci dlouhodobého výzkumu používají tento program a pomocí počtu detekovaných sekundárních šupin odhadují pohlaví nedospělých ještěrek [49]. Otestovali jej na odlišné datové sadě, než která byla použita v této práci, což dokazuje, že je program robustní a funkční. Program urychluje identifikaci pohlaví, omezuje lidský faktor při počítání šupin, a tím zpřesňuje výsledky. Díky programu není nutné pro určení pohlaví ještěrek používat analýzu DNA, což vědcům šetří finanční prostředky a čas.

Ještěrka obecná patří k plazům, u kterých pohlaví nově vylíhnutých jedinců závisí na teplotě prostředí. Změny teplot v lokalitách, které vědci z ÚBO AV ČR v.v.i. a PdF MUNI dlouhodobě monitorují, se proto mohou projevit v poměru pohlaví nedospělých ještěrek, a tím i v počtu jejich populace. Navržený program pomůže biologům porozumět těmto změnám.

Výsledky diplomové práce ukázaly, že Centroid detektor, který byl původně určený k počítání lidí nastupujících do metra, je vhodný i na jiné aplikace, jako je počítání specifické řady šupin na břicho ještěrky. Centroid detektor se dá využít pro určení pozice homogenních objektů, které jsou v obrazových datech velmi blízko u sebe.



## SEZNAM POUŽITÉ LITERATURY

- [1] AVERY, R. A. Age-structure and longevity of Common lizard (*Lacerta vivipara*) populations. *Journal of Zoology* [online]. 1975, 176(4), 555-558 [cit. 2021-10-27]. ISSN 0952-8369. Dostupné z: doi:10.1111/j.1469-7998.1975.tb03221.x.
- [2] ROVATSOS, Michail, Jasna VUKIĆ, Agata MRUGAŁA, Grzegorz SUWALA, Petros LYMBERAKIS a Lukáš KRATOCHVÍL. Little evidence for switches to environmental sex determination and turnover of sex chromosomes in lacertid lizards. *Scientific Reports* [online]. 2019, 9(1) [cit. 2021-10-14]. ISSN 2045-2322. Dostupné z: doi:10.1038/s41598-019-44192-5.
- [3] EPLANOVA, Galina V. a Evgeny S. ROITBERG. Sex identification of juvenile sand lizards, *Lacerta agilis* using digital images. *Amphibia-Reptilia* [online]. 2015, 36(3), 215-222 [cit. 2021-10-10]. ISSN 0173-5373. Dostupné z: doi:10.1163/15685381-00002996.
- [4] SMOLINSKÝ, Radovan. *Interní sbírka fotografií ještěrek obecných*. Brno: Pedagogická fakulta Masarykovy univerzity, 2021.
- [5] DIJKSTRA, Klaas, Jaap VAN DE LOOSDRECHT, Waatze A. AT SMA, Lambert R.B. SCHOMAKER a Marco A. WIERING. CentroidNetV2: A hybrid deep neural network for small-object segmentation and counting. *Neurocomputing* [online]. 2021, 423, 490-505 [cit. 2022-02-09]. ISSN 09252312. Dostupné z: doi:10.1016/j.neucom.2020.10.075.
- [6] HSIEH, Meng-Ru, Yen-Liang LIN a Winston H. HSU. Drone-Based Object Counting by Spatially Regularized Regional Proposal Network. *2017 IEEE International Conference on Computer Vision (ICCV)* [online]. IEEE, 2017, 2017, 4165-4173 [cit. 2022-02-09]. ISBN 978-1-5386-1032-9. Dostupné z: doi:10.1109/ICCV.2017.446.
- [7] MEZEI, Sergiu a Adrian DARABANT. A computer vision approach to object tracking and counting. *Studia Universitatis Babeş-Bolyai, Informatica* [online]. 2010, 55(1) [cit. 2022-02-09]. Dostupné z: <https://linkinghub.elsevier.com/retrieve/pii/S0925231220316647>.
- [8] DIJKSTRA, K., J. VAN DE LOOSDRECHT, L. R. B. SCHOMAKER a M. A. WIERING. CentroidNet: A Deep Neural Network for Joint Object Localization and Counting. *Machine Learning and Knowledge Discovery in Databases* [online]. Cham: Springer International Publishing, 2019, 2019-01-18, 585-601 [cit. 2022-02-09]. Lecture Notes in Computer Science. ISBN 978-3-030-10996-7. Dostupné z: doi:10.1007/978-3-030-10997-4\_36.

- [9] MITCHELL, Tom M. *Machine learning*. Boston: McGraw-Hill, 1997. ISBN 00-704-2807-7.
- [10] MCCULLOCH, Warren S. a Walter PITTS. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* [online]. 1943, 5(4), 115-133 [cit. 2022-02-05]. ISSN 0007-4985. Dostupné z: doi:10.1007/BF02478259.
- [11] ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* [online]. 1958, 65(6), 386-408 [cit. 2022-02-05]. ISSN 1939-1471. Dostupné z: doi:10.1037/h0042519.
- [12] Mish: A self regularized non-monotonic activation function. *CoRR* [online]. 2019, 1-14 [cit. 2022-02-05]. Dostupné z: <https://doi.org/10.48550/arXiv.1908.08681>.
- [13] WERBOS, Paul. *Beyond regression: New tools for prediction and analysis in the behavior science* [online]. Harvard, 1974 [cit. 2022-02-06]. Dostupné z: [https://books.google.cz/books/about/Beyond\\_Regression.html?id=z81XmgEACAAJ&redir\\_esc=y](https://books.google.cz/books/about/Beyond_Regression.html?id=z81XmgEACAAJ&redir_esc=y). Disertační práce. Harvard University.
- [14] WERBOS, Paul J. Applications of advances in nonlinear sensitivity analysis. *System Modeling and Optimization* [online]. Berlin/Heidelberg: Springer-Verlag, 1982, 762-770 [cit. 2022-02-06]. Lecture Notes in Control and Information Sciences. ISBN 3-540-11691-5. Dostupné z: doi:10.1007/BFb0006203.
- [15] RUMELHART, David E., Geoffrey E. HINTON a Ronald J. WILLIAMS. Learning representations by back-propagating errors. *Nature* [online]. 1986, 323(6088), 533-536 [cit. 2022-02-09]. ISSN 0028-0836. Dostupné z: doi:10.1038/323533a0.
- [16] KINGMA, Diederik P. a Jimmy BA. Adam: A method for stochastic optimization. *International Conference on Learning Representations* [online]. 2014, 1-15 [cit. 2022-05-09]. Dostupné z: doi:<https://doi.org/10.48550/arXiv.1412.6980>.
- [17] HEBB, Donald Olding. *The organization of behavior: a neuropsychological theory*. Science editions, 1949. ISBN 978-0805843002.
- [18] GONZALEZ, Rafael C. a Richard E. WOODS. *Digital image processing*. Fourth edition. New York: Pearson, [2018]. ISBN 978-013-3356-724.
- [19] HUBEL, D. H. a T. N. WIESEL. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology* [online]. 1959, 148(3), 574-591 [cit. 2022-02-07]. ISSN 00223751. Dostupné z: doi:10.1113/jphysiol.1959.sp006308.



- [20] FUKUSHIMA, Kunihiro. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* [online]. 1980, 36(4), 193-202 [cit. 2022-02-12]. ISSN 0340-1200. Dostupné z: doi:10.1007/BF00344251.
- [21] SHORTEN, Connor a Taghi M. KHOSHGOFTAAR. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* [online]. 2019, 6(1), 1-48 [cit. 2022-02-10]. ISSN 2196-1115. Dostupné z: doi:10.1186/s40537-019-0197-0.
- [22] BOYAT, Ajay Kumar a Brijendra Kumar JOSHI. A Review Paper: Noise Models in Digital Image Processing. *Signal Image Processing : An International Journal* [online]. 2015, 6(2), 63-75 [cit. 2022-02-15]. ISSN 22293922. Dostupné z: doi:10.5121/sipij.2015.6206.
- [23] SINGH, P. a Raj Shree PANDEY. Speckle noise: Modeling and implementation. *International Science Press* [online]. 2016, 9(17), 8717-8727 [cit. 2022-03-16]. Dostupné z: [https://www.researchgate.net/publication/312061215\\_Speckle\\_noise\\_Modeling\\_and\\_implementation](https://www.researchgate.net/publication/312061215_Speckle_noise_Modeling_and_implementation).
- [24] BERZINS, Zigmars. Girl, dog, pet, friendship companion. In: *Pixabay* [online]. 2019 [cit. 2022-03-09]. Dostupné z: <https://pixabay.com/photos/girl-dog-pet-friendship-companion-5623231/>.
- [25] LU, D. a Q. WENG. A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing* [online]. 2007, 28(5), 823-870 [cit. 2022-03-10]. ISSN 0143-1161. Dostupné z: doi:10.1080/01431160600746456.
- [26] ZOU, Zhengxia, Zhenwei SHI, Yuhong GUO a Jieping YE. Object Detection in 20 Years: A survey. *IEEE TPAMI* [online]. 2019, 1-39 [cit. 2022-02-12]. Dostupné z: doi:<https://doi.org/10.48550/arXiv.1905.05055>.
- [27] ZHAO, Zhong-Qiu, Peng ZHENG, Shou-Tao XU a Xindong WU. Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems* [online]. 2019, 30(11), 3212-3232 [cit. 2022-02-12]. ISSN 2162-237X. Dostupné z: doi:10.1109/TNNLS.2018.2876865.
- [28] ZHENG, Zhaohui, Ping WANG, Dongwei REN, Wei LIU, Rongguang YE, Qinghua HU a Wangmeng ZUO. Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation. *IEEE Transactions on Cybernetics* [online]. 2021, 1-13 [cit. 2022-02-12]. ISSN 2168-2267. Dostupné z: doi:10.1109/TCYB.2021.3095305.

- [29] DAVIS, Jesse a Mark GOADRICH. The relationship between Precision-Recall and ROC curves. *Proceedings of the 23rd international conference on Machine learning - ICML '06* [online]. New York, New York, USA: ACM Press, 2006, 2006, 233-240 [cit. 2022-03-12]. ISBN 1595933832. Dostupné z: doi:10.1145/1143844.1143874.
- [30] GIRSHICK, Ross, Jeff DONAHUE, Trevor DARRELL a Jitendra MALIK. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition* [online]. IEEE, 2014, 2014, 580-587 [cit. 2022-03-12]. ISBN 978-1-4799-5118-5. Dostupné z: doi:10.1109/CVPR.2014.81.
- [31] GIRSHICK, Ross. Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)* [online]. IEEE, 2015, 2015, 1440-1448 [cit. 2022-03-12]. ISBN 978-1-4673-8391-2. Dostupné z: doi:10.1109/ICCV.2015.169.
- [32] REN, Shaoqing, Kaiming HE, Ross GIRSHICK a Jian SUN. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2017, 39(6), 1137-1149 [cit. 2022-03-12]. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2016.2577031.
- [33] REDMON, Joseph, Santosh DIVVALA, Ross GIRSHICK a Ali FARHADI. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 2016, 2016, 779-788 [cit. 2022-03-12]. ISBN 978-1-4673-8851-1. Dostupné z: doi:10.1109/CVPR.2016.91.
- [34] BOCHKOVSKIY, Alexey, Chien-Yao WANG a Hong-Yuan Mark LIAO. *YOLOv4: Optimal Speed and Accuracy of Object Detection* [online]. 2020, 1-17 [cit. 2022-03-12]. Dostupné z: doi:https://doi.org/10.48550/arXiv.2004.10934.
- [35] MINAEI, Shervin, Yuri Y. BOYKOV, Fatih PORIKLI, Antonio J PLAZA, Nasser KEHTARNAVAZ a Demetri TERZOPOULOS. Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 1-1 [cit. 2022-03-20]. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2021.3059968.
- [36] CRESWELL, Antonia, Kai ARULKUMARAN a Anil A. BHARATH. On denoising autoencoders trained to minimise binary cross-entropy. *Imperial College London* [online]. 2017, 1-8 [cit. 2022-03-20]. Dostupné z: doi:https://doi.org/10.48550/arXiv.1708.08487.

- [37] BADRINARAYANAN, Vijay, Alex KENDALL a Roberto CIPOLLA. Seg-Net: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2017, 39(12), 2481-2495 [cit. 2022-03-20]. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2016.2644615.
- [38] MILLETARI, Fausto, Nassir NAVAB a Seyed-Ahmad AHMADI. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. *2016 Fourth International Conference on 3D Vision (3DV)* [online]. IEEE, 2016, 2016, 565-571 [cit. 2022-03-21]. ISBN 978-1-5090-5407-7. Dostupné z: doi:10.1109/3DV.2016.79.
- [39] RONNEBERGER, Olaf, Philipp FISCHER a Thomas BROX. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* [online]. Cham: Springer International Publishing, 2015, 2015-11-18, 234-241 [cit. 2022-03-20]. Lecture Notes in Computer Science. ISBN 978-3-319-24573-7. Dostupné z: doi:10.1007/978-3-319-24574-4\_28.
- [40] DOLEŽEL, Petr, Pavel ŠKRABÁNEK, Dominik ŠTURSA, Bruno Baruque ZANON, Hector Cogollos ADRIAN a Pavel KŘÍDA. Centroid based person detection using pixelwise prediction of the position. *Journal of Computational Science*. 2022. V revizním řízení.
- [41] DASARATHY, B.V. a B.V. SHEELA. A composite classifier system design: Concepts and methodology. *Proceedings of the IEEE* [online]. 1979, 67(5), 708-713 [cit. 2022-03-25]. ISSN 0018-9219. Dostupné z: doi:10.1109/PROC.1979.11321.
- [42] DONG, Xibin, Zhiwen YU, Wenming CAO, Yifan SHI a Qianli MA. A survey on ensemble learning. *Frontiers of Computer Science* [online]. 2020, 14(2), 241-258 [cit. 2022-03-25]. ISSN 2095-2228. Dostupné z: doi:10.1007/s11704-019-8208-z.
- [43] DIETTERICH, Thomas G. Ensemble Methods in Machine Learning. *Multiple Classifier Systems* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, 2000-12-1, s. 1-15 [cit. 2022-03-25]. Lecture Notes in Computer Science. ISBN 978-3-540-67704-8. Dostupné z: doi:10.1007/3-540-45014-9\_1.
- [44] ZHANG, Cha a Yunqian MA, ed. *Ensemble Machine Learning* [online]. Boston, MA: Springer US, 2012 [cit. 2022-03-25]. ISBN 978-1-4419-9325-0. Dostupné z: doi:10.1007/978-1-4419-9326-7.

- [45] DRAČKOVÁ, Tereza, Radovan SMOLINSKÝ, Zuzana HIADLOVSKÁ, Matej DOLINAY a Natália MARTÍNKOVÁ. Quantifying colour difference in animals with variable patterning. *Journal of Vertebrate Biology* [online]. 2020, 69(4) [cit. 2021-10-15]. ISSN 2694-7684. Dostupné z: doi:10.25225/jvb.20029.
- [46] DE BOEVER, Henri. CSC462-distributed-image-processing. In: *GitHub* [online]. 2019 [cit. 2022-03-12]. Dostupné z: <https://github.com/HDeBoever/CSC462-distributed-image-processing/blob/master/yolo-object-detection/yolo.py>.
- [47] TECHZIZOU. Train a custom YOLOv4 object detector on Windows. In: *TechZizou* [online]. 2021 [cit. 2022-03-10]. Dostupné z: <https://techzizou.com/train-a-custom-yolov4-object-detector-on-windows/>.
- [48] BHATTIPROLU, Sreenivas. Python for Microscopists and other image processing enthusiasts. In: *GitHub* [online]. 2021 [cit. 2022-03-15]. Dostupné z: [https://github.com/bnsreenu/python\\_for\\_microscopists](https://github.com/bnsreenu/python_for_microscopists).
- [49] SMOLINSKÝ, Radovan, Zuzana HIADLOVSKÁ, Štěpán MARŠALA, Pavel ŠKRABÁNEK, MARTÍNKOVÁ a Natália MARTÍNKOVÁ. High predation risk decimates survival during the reproduction season. *BioRxiv* [online]. Cold Spring Harbor Laboratory, 2022 [cit. 2022-04-02]. Dostupné z: doi:10.1101/2022.03.31.486539.

## SEZNAM ZKRATEK

<b>ÚBO AV ČR v.v.i.</b>	Ústav biologie obratlovců Akademie věd ČR, veřejná výzkumná instituce
<b>PdF MUNI</b>	Pedagogická fakulta Masarykovy univerzity
<b>SGD</b>	ztrátová funkce zvaná stochastický gradientní sestup (stochastic gradient descent)
<b>RGB</b>	barevný prostor (Red Green Blue)
<b>IoU</b>	ztrátová funkce zvaná Intersection over Union
<b>mAP</b>	metrika zvaná mean Average Precision
<b>AP</b>	metrika zvaná Average Precision
<b>TP</b>	pravdivě pozitivní detekce
<b>FP</b>	nepravdivě pozitivní detekce
<b>FN</b>	nepravdivě negativní detekce
<b>R-CNN</b>	Region-based Convolutional Neural Network
<b>CE</b>	křížová entropie (cross entropy)
<b>BCE</b>	binární křížová entropie (Binary Cross Entropy)
<b>YOLO</b>	detektor objektů (You Only Look Once)
<b>YOLOv4</b>	detektor objektů (You Only Look Once verze 4)
<b>C</b>	sada konvolučních vrstev v síti U-Net
<b>P</b>	vrstva sjednocení v síti U-Net
<b>U</b>	vrstva rozšíření v síti U-Net



## SEZNAM SYMBOLŮ

$\hat{z}$	výstupní hodnota neuronu
$a(\cdot)$	aktivační funkce neuronu
$J_u$	počet vstupů do neuronu
$w$	váha spojení mezi neurony
$u$	hodnota vstupního signálu do neuronu
$b$	bias neuronu
$v$	suma biasu a součinu vah spojení $w$ se vstupy $u$
$i$	pomocná proměnná
$a_l$	parametr záporné části aktivační funkce leaky
$C(\cdot)$	ztrátová funkce
$z$	očekávaná hodnota na výstupu neuronu (ground truth)
$\mu$	learning rate
$\alpha$	konstanta momentové věty
$\hat{m}_w$	funkce hybnosti a derivace ztrátové funkce $C$ podle váhy $w$
$\hat{v}_w$	funkce hybnosti a derivace ztrátové funkce $C$ podle váhy $w$
$\epsilon$	nenulový parametr optimalizačního algoritmu Adam
$M$	horizontální rozměr obrazové funkce [pixel]
$N$	vertikální rozměr obrazové funkce [pixel]
$f(\cdot)$	obrazová funkce
$x$	horizontální souřadnice obrazové funkce [pixel]
$y$	vertikální souřadnice obrazové funkce [pixel]
$h(\cdot)$	konvoluční maska
$M_h$	horizontální rozměr konvoluční masky $h(\cdot)$ [pixel]
$N_h$	vertikální rozměr konvoluční masky $h(\cdot)$ [pixel]
$g(\cdot)$	konvoluční vrstva
$j$	pomocná proměnná
$M_g$	horizontální rozměr konvoluční vrstvy $g(\cdot)$ [pixel]
$N_g$	vertikální rozměr konvoluční vrstvy $g(\cdot)$ [pixel]
$M_p$	vertikální rozměr podoblasti vrstvy sjednocení (pooling) [pixel]
$N_p$	horizontální rozměr podoblasti vrstvy sjednocení (pooling) [pixel]
$M_{us}$	vertikální rozměr podoblasti vrstvy rozšíření (up sample) [pixel]
$N_{us}$	horizontální rozměr podoblasti vrstvy rozšíření (up sample) [pixel]
$x'$	horizontální souřadnice augmentovaného obrazu [pixel]
$y'$	vertikální souřadnice augmentovaného obrazu [pixel]
$\mathcal{T}(\cdot)$	obrazová transformační funkce
$T(\cdot)$	matice afinní transformace
$\theta$	úhel natočení afinní transformace rotace [rad]

$t_x$	posunutí obrazu afinní transformace translace ve směru osy x [pixel]
$t_y$	posunutí obrazu afinní transformace translace ve směru osy y [pixel]
$s_x$	parametr zkosení obrazu afinní transformace ve směru osy x
$s_y$	parametr zkosení obrazu afinní transformace ve směru osy y
$c_x$	parametr škálování obrazu afinní transformace ve směru osy x
$c_y$	parametr škálování obrazu afinní transformace ve směru osy y
$P(\cdot)$	rozdělení pravděpodobnosti
$q$	střední hodnota Gaussova rozdělení pravděpodobnosti
$\sigma$	směrodatná odchylka Gaussova rozdělení pravděpodobnosti
$P_s$	pravděpodobnost šumu typu sůl
$P_p$	pravděpodobnost šumu typu pepř
$n$	třída
$I$	vstupní obraz
$X$	množina všech možných tříd
$x_c$	jistota (confidence) výstupu konvoluční neuronové sítě
$\hat{\cdot}$	odhad výstupních veličin konvoluční neuronové sítě
$L$	výstupní množina klasifikátoru obrazu
$l$	dvojice $n$ a $\hat{x}_c$ v množině $\hat{L}$
$k$	uspořádaná šestice definující bounding-box detekovaného objektu
$K$	množina šestic $k$ , výstup detektoru objektů
$x_{ul}$	souřadnice x levého horního bodu bounding-boxu
$y_{ul}$	souřadnice y levého horního bodu bounding-boxu
$x_{br}$	souřadnice x pravého dolního bodu bounding-boxu
$y_{br}$	souřadnice y pravého dolního bodu bounding-boxu
$C_{IoU}(\cdot)$	ztrátová funkce Intersection over Union (IoU)
$A_p$	plocha ohraničená odhadovaným bounding-boxem detekovaného objektu
$A_{gt}$	plocha ohraničená očekávaným bounding-boxem objektu
$t_{IoU}$	práh hodnoty funkce $IoU$
$ \cdot $	počet prvků v dané množině
$P$	hodnota přesnosti (precision)
$R$	hodnota recall
$J_{PR}$	počet prahových hodnot $t_{IoU}$
$P_A$	hodnota metriky Average Precision (AP)
$P_{mA}$	hodnota metriky mean Average Precision (mAP)
$AP_n$	hodnota AP třídy $n$
$S$	počet segmentů čtvercové mřížky metody YOLO
$t_{bb}$	prahová hodnota metody non-maxima suppression
$u$	výstupní segmentační mapa segmentace obrazu třídy $n$
$U$	množina výstupních map $u$ , výstup segmentace obrazu



$E_C$	hodnota křížové entropie CE
$E_{BC}$	hodnota binární křížové entropie BCE
$\Phi$	množina hodnot pixelů vstupního obrazu $I$
$\Gamma$	množina souřadnic středů detekovaných gradientů výstup centroid detektoru
$x_{cd}$	souřadnice x středu gradientu centroid detektoru
$y_{cd}$	souřadnice y středu gradientu centroid detektoru
$\mathcal{L}$	lokalizační mapa gradientů
$B(\cdot)$	maxfiltr
$S_{xy}$	soubor prostorových souřadnic v masce maxfiltru $B(\cdot)$
$M_B$	horizontální rozměr masky maxfiltru $B(\cdot)$ [pixel]
$N_B$	vertikální rozměr masky maxfiltru $B(\cdot)$ [pixel]
$B_1(\cdot)$	maska stejných hodnot $B(\cdot)$ a $\hat{\mathcal{L}}$
$\Omega(\cdot)$	maska mapující místa, kde v $\hat{\mathcal{L}}$ nejsou gradienty
$\Omega_{\ominus}(\cdot)$	eroze masky $\Omega(\cdot)$
$\varepsilon$	okolí eroze
$M_{\varepsilon}$	horizontální rozměr okolí eroze [pixel]
$N_{\varepsilon}$	vertikální rozměr okolí eroze [pixel]
$B_{\Omega}(\cdot)$	logický XOR masek $B_1(\cdot)$ a $\Omega_{\ominus}(\cdot)$
$\Xi(\cdot)$	mapa středů detekovaných gradientů
$t_m$	práh citlivosti centroid counterpointu
$S_{xyn}$	rozměry $\Xi(\cdot)$
$Y$	unifikovaný výřez ještěrky
$\Psi_L$	množina pozic sekundárních šupin na levé straně ventrální strany ještěrky
$\Psi_P$	množina pozic sekundárních šupin na pravé straně ventrální strany ještěrky
$p_{\emptyset}$	průměr euklidovských vzdáleností sousedních bodů množiny $\hat{\Gamma}$
$m(\cdot)$	funkce určující hodnoty pixelů gradientu
$r$	průměr gradientu v mapě [pixel]
$d$	vzdálenost od středu gradientu [pixel]



## SEZNAM OBRÁZKŮ

1	Vyznačení primárních šupin (zeleně), sekundárních šupin (červeně), límce (černě) a šupin s femorálními póry (modře) na ventrální straně těla ještěrky obecné. Převzato a upraveno z [4]. . . . .	17
2	Detail (červeně) posledních ventrálních sekundárních šupin. Převzato a upraveno z [4]. . . . .	17
3	Model umělého neuronu odpovídající rovnici (1). . . . .	22
4	Různé typy aktivačních funkcí. . . . .	23
5	Příklad umělé neuronové sítě s jednou skrytou vrstvou. . . . .	23
6	Ukázka souřadného systému používaného v softwaru OpenCV pro zpracování obrazu. . . . .	26
7	Zero-padding obrazové funkce (vlevo) pro dvourozměrnou konvoluci konvoluční maskou $h(\cdot)$ o rozměrech $M_h$ a $M_h = 3 \times 3$ . . . . .	28
8	Ukázka prvního kroku diskrétní dvourozměrné konvoluce konvoluční vrstvy rozšířené zero-paddingem (viz obr. 7) na rozměry $6 \times 6$ konvoluční maskou $3 \times 3$ . . . . .	28
9	Ukázka dvou variant vrstvy sjednocení. Vlevo je původní obrazová funkce o rozměrech $M \times N = 4 \times 4$ , vpravo jsou obrazové funkce po sjednocení max-pooling mřížkou $M_p \times N_p = 2 \times 2$ a average-pooling mřížkou $M_p \times N_p = 2 \times 2$ . Barevně jsou vyznačeny skupiny sjednocených pixelů. . . . .	29
10	Ukázka vrstvy rozšíření. Vlevo je původní obrazová funkce o rozměrech $M \times N = 2 \times 2$ , vpravo je obrazová funkce po rozšíření skupinou pixelů o rozměrech $M_{us} \times N_{us} = 2 \times 2$ . Barevně jsou vyznačeny skupiny rozšířených pixelů. . . . .	29
11	Ukázka rotace o kladný úhel $\theta$ . . . . .	31
12	Ukázka translace o zápornou hodnotu $t_x$ a kladnou hodnotu $t_y$ . . . . .	31
13	Ukázka horizontálního převrácení. . . . .	32
14	Ukázka zkosení o kladný parametr $s_x$ . . . . .	32
15	Ukázka geometrické transformace škálování o kladný parametr $c_x$ . . . . .	32
16	Ukázka šumů. Gaussovský šum s hodnotami $q = 0$ a $\sigma = 20$ . Speckle s hodnotami $q = 0$ a $\sigma = 1$ . Sůl a pepř s hodnotami $P_s = 0,05$ a $P_p = 0,05$ . Fotografie na obrázku byly převzaty a upraveny z [24]. . . . .	33
17	Ukázka klasifikátoru, který klasifikuje vstupní obraz $I$ třídy pes. Fotografie na obrázku byla převzata a upravena z [24]. . . . .	34

18	Ukázka detektoru objektů. Ve vstupním obraze $I$ jsou detekovány objekty třídy pes (červený bounding-box) a člověk (zelený bounding-box). Fotografie byly převzaty a upraveny z [24]. . . . .	35
19	Schéma procesů detektoru YOLO. Nalevo je vstupní obraz rozdělený na segmenty, napravo jsou v obraze vyznačené detekované bounding-boxy po non-maxima suppression. Převzato a upraveno z [34]. . . . .	37
20	Ukázka segmentace obrazu tříd pes a člověk. Napravo jsou dvě výstupní mapy. Červenou barvou je mapována třída pes a zelenou barvou třída člověk. Fotografie na obrázku byla převzata a upravena z [24].	38
21	Ukázka segmentace obrazu pomocí sítě U-Net. Vpravo je vstupní obraz, vlevo je výstupní mapa třídy buňka, na které jsou bílou barvou namapovány buňky. Černá barva jsou místa, kde U-Net buňky nedetekoval. [39]. . . . .	39
22	Architektura sítě U-Net o pěti úrovních a 64 konvolučními maskami v konvolučních vrstvách první úrovně. Síť přijímá černobílou fotografii o rozměrech $572 \times 572 \times 1$ pixelů a vrací dvě mapy $\hat{u}$ pro každou třídu $n$ o rozměrech $388 \times 388 \times 2$ pixelů. Převzato a upraveno z [39]. . . . .	40
23	Ukázka Centroid detektoru, který přijímá vstupní obraz $I$ a vrací množinu $\hat{\Gamma}$ . Fotografie na obrázku byla převzata a upravena z [24]. . .	41
24	Ukázka generátoru lokalizačních map, který vytváří odhad lokalizační mapy $\hat{\mathcal{L}}$ s rozostřeným gradientem uprostřed dvou objektů. Lokalizační mapy různých tříd jsou spojeny a barevně rozlišeny. Červený gradient lokalizuje střed objektu třídy pes a zelený gradient lokalizuje střed objektu třídy člověk. Fotografie na obrázku byla převzata a upravena z [24]. . . . .	42
25	Ukázka chodu Centroid counterpointu. . . . .	42
26	Příklad fotografie ještěrky obecné [4]. . . . .	45
27	Znázornění navrženého systému strojového vidění. Šipky zobrazují směr toku dat mezi operacemi v obdélnících. Nad šipkami jsou proměnné reprezentující data, pod šipkami jsou příklady obrázků, které tato data vizualizují. Fotografie na obrázku byly převzaty a upraveny z [4]. . . . .	47
28	Výstup YOLOv4. Vlevo jsou zobrazeny detekované objekty před použitím non-maxima suppression a vpravo jsou objekty po použití této metody. Fotografie byly převzaty a upraveny z [4]. . . . .	48
29	Unifikovaný výřez $Y$ vstupní fotografie $I$ z obr. 26 o rozměrech $512 \times 512$ pixelů. Velikost a umístění výřezu založena na výstupech YOLOv4 detektoru. Černý pruh v horní části obrazu vznikl doplněním původního obrazu černou. Fotografie byla převzata a upravena z [4]. . . . .	50

30	Architektura modifikované sítě U-Net o pěti úrovních a 64 konvolučními maskami v konvolučních vrstvách první úrovně. Síť přijímá výřez $Y$ o rozměrech $512 \times 512 \times 3$ pixelů a vrací lokalizační mapu $\hat{\mathcal{L}}$ o rozměrech $512 \times 512 \times 1$ pixelů s hodnotami 0 až 255. . . . .	51
31	Příklad lokalizační mapy $\hat{\mathcal{L}}$ o rozměrech $512 \times 512$ výřezu $Y$ z obr. 29 s hodnotami 0 (černá) až 255 (bílá). Lokální maxima jsou pozice detekovaných šupin. . . . .	52
32	Schéma použití souborového učení tří zvolených modelů. . . . .	53
33	Grafické znázornění Centroid counterpointu pro mapu z obr. 31. Rovnice pro zobrazené obrazové operace jsou popsány v podkapitole 3.8.2. Hodnota prahu $t_m = 0,1$ a parametr <code>MINIMUM_DISTANCE</code> byl zadán 2. . . . .	54
34	Ukázka detekovaných souřadnic středů šupin množiny $\hat{\Gamma}$ v lokalizační mapě gradientů $\hat{\mathcal{L}}$ z obr. 31. Červené křížky označují nalezené středy lokálních maxim gradientů. V levém horním rohu je znázorněn souřadný systém. . . . .	55
35	Vizualizace rozdělení souřadnic středů detekovaných sekundárních šupin množiny $\hat{\Gamma}$ na levou $\Psi_L$ (modré tečky) a pravou $\Psi_P$ (červené tečky) řadu. Nalevo jsou detekované šupiny zobrazeny v lokalizační mapě gradientů $\hat{\mathcal{L}}$ (viz obr. 31) a napravo jsou zobrazeny ve výřezu $Y$ (viz obr. 29). Fotografie na obrázku byla převzata a upravena z [4]. . . . .	56
36	Ukázka vyřazených fotografií [4]. Na levé fotografii nejsou viditelné všechny šupiny a na pravé fotografii není ještěřka vyfocená z ventrální strany. . . . .	57
37	Štítkování břicha, ocasu a hlavy fotografie $I$ v programu LabelImg. Fotografie na obrázku byla převzata a upravena z [4]. . . . .	58
38	Navržený štítkovací program. Nalevo je prostředí pro označování šupin myši (modrá kolečka jsou šupiny levé sekundární řady šupin, červená kolečka jsou šupiny pravé sekundární řady šupin). Napravo je vizualizace očekávané lokalizační mapy $\mathcal{L}$ s hodnotami 0 (černá) až 255 (bílá). V levém horním rohu je barvou označená štítkovaná třída, první číslo zobrazuje poloměr gradientu v pixelech, druhé číslo pořadí výřezu $Y$ . . . . .	59
39	Průběh trénování modelu YOLOv4. Červeně je průběh mAP (mean Average Precision) a modře hodnota ztrátové funkce IoU (Intersection over Union). Váhy se neukládaly po 8 000 iteracích, kde $mAP = 92,53\%$ měla nejvyšší hodnotu. . . . .	61
40	Struktura adresáře <code>yolov4/</code> , která je nutná pro natrénování modelu YOLOv4 systémem darknet. Adresář je dostupný v příloze A ve složce <code>google_colab/</code> . . . . .	61

41	Ukázka augmentace šumem obr. 29. Gaussovský šum s hodnotami $q = 0$ a $\sigma = 10$ . Speckle s hodnotami $q = 0$ a $\sigma = 0,7$ . Sůl a pepř s hodnotami $P_s = 0,05$ a $P_p = 0,05$ . Fotografie na obrázku byly převzaty a upraveny z [4]. . . . .	62
42	Vlevo je ukázka lineárně klesajícího gradientu popsaného rovnicí (48) a vpravo je nelineárně klesající gradient popsaný rovnicí (49). Červeně jsou popsány hodnoty vybraných pixelů. Oba gradienty mají poloměr $r = 7$ pixelů. . . . .	64
43	Struktura adresáře <code>unet/</code> , která je nutná pro natrénování modelů U-Net. Adresář je umístěn ve kmenové složce Google Disku. Soubor <code>images_data.pickle</code> je na obrázku jen pro ilustraci a není součástí přílohy A. . . . .	65
44	Průběhy trénování vybraných modelů 1, 2 a 3. . . . .	68
45	Výstupní lokalizační mapy $\hat{\mathcal{L}}$ vybraných modelů 1, 2 a 3 (zleva). Vstupní výřez $Y$ je na obr. 29. . . . .	68
46	Ukázka běhu lokální verze programu pro Windows. . . . .	70
47	Ukázka běhu cloudové verze programu. . . . .	71
48	Časová analýza průměrné délky zpracování jedné fotografie všech vstupních formátů jak lokální (vlevo), tak cloudové (vpravo) verze programu. Celkový čas výpočtu je barevně rozdělen do částí systému strojového vidění. . . . .	73
49	Struktura adresáře <code>priloha_A/</code> . . . . .	99
50	Struktura adresáře <code>priloha_B/</code> . . . . .	101

## SEZNAM TABULEK

1	Přehled nejdůležitějších vstupních parametrů pro trénování sítě YOLOv4. ....	60
2	Popis vybraných modelů. ....	63
3	Parametry nastavené během trénování vybraných modelů. ....	68
4	Hodnoty parametrů nastavené během experimentu. ....	71
5	Četnosti odchylek detekovaného počtu šupin od skutečného počtu na testovací sadě 77 fotografií. ....	72
6	Statistické zhodnocení odchylek detekovaného počtu šupin od skutečného počtu na testovací sadě 77 fotografií. ....	72





## SEZNAM PŘÍLOH

A	Sada skriptů k natrénování modelů . . . . .	99
B	Programy . . . . .	101



## A Sada skriptů k natrénování modelů

Příloha A obsahuje sadu skriptů k natrénování modelů konvolučních neuronových sítí YOLOv4 a U-Net. Návod k obsluze je v souboru `README.md`. Jde o adresář `priloha_A/` se strukturou, která je znázorněná na obr. 49. Příloha je dostupná v archivu závěrečné práce v souboru `StepanMarsala_200203_DP_2022_UAI.zip`.

```
priloha_A/  
|__centroid_counterpoint.py  
|__configure.py  
|__cut_body.py  
|__labeling.py  
|__pickle2csv.py  
|__prepare.py  
|__README.md  
|__requirements.txt  
|__thesis_evaluation.py  
|__dataset_raw/  
|__models/  
|__google_colab/  
|__unet_jupyter_train.ipynb  
|__yolov4_jupyter_train.ipynb  
|__dataset_raw/  
|__unet/  
|    |__models/  
|    |__test_data_images/  
|    |__train_data_images/  
|__yolov4/  
|    |__obj.data  
|    |__obj.names  
|    |__obj.zip  
|    |    |__classes.txt  
|    |__process.py  
|    |__yolov4-custom.cfg  
|    |__training/
```

Obr. 49: Struktura adresáře `priloha_A/`.



## B Programy

Příloha B obsahuje lokální verzi výsledného programu pro Windows a cloudovou verzi. Návod k obsluze je v souborech `README.md` jednotlivých verzí. Jde o adresář `priloha_B/` se strukturou, která je znázorněná na obr. 50. Příloha je dostupná v archivu závěrečné práce v souboru `StepanMarsala_200203_DP_2022_UAI.zip`.

```
priloha_A/  
|__cloud/  
|  |__README.md  
|  |__ScalesCounter.ipynb  
|  |__ScalesCounter/  
|      |__main_program.py  
|      |__scales_counter.py  
|      |__yolov4-custom.cfg  
|      |__input_images/  
|      |__models/  
|__local/  
|  |__config.yaml  
|  |__main.py  
|  |__README.md  
|  |__requirements.txt  
|  |__scales_counter.py  
|  |__setup.py  
|  |__yolov4-custom.cfg  
|  |__images_input/  
|  |__models/
```

Obr. 50: Struktura adresáře `priloha_B/`.