



## **STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST**

**Obor č. 18: Informatika**

# **Segmentace obrazů členovců pomocí neuronových sítí**

**Matěj Pekár**

**Jihomoravský kraj, Brno 2021**

# **STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST**

**Obor č. 18: Informatika**

## **Segmentace obrazů členovců pomocí neuronových sítí**

**Segmentation of arthropod images  
using neural networks**

**Autor: Matěj Pekár**

**Škola: Gymnázium Brno, Vídeňská, příspěvková organizace, Vídeňská 55/47, 639 00 Brno**

**Kraj: Jihomoravský**

**Konzultant: doc. RNDr. Petr Matula, Ph.D.**

## **Prohlášení**

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Brně dne .....

Matěj Pekár

## **Poděkování**

Na tomto místě bych rád poděkoval svému mentorovi doc. RNDr. Petru Matulovi, Ph. D. za obětavou pomoc, čas, který mi věnoval a nadšení, které sdílel pro tuto problematiku. Také děkuji svému otci Prof. Mgr. Stanislavu Pekárovi, Ph. D. za poskytnutí referenčních dat.

Tato práce byla vypracována za finanční podpory Jihomoravského kraje.



**jihomoravský kraj**

## Anotace

Mimeze je vzhledové nebo zvukové napodobení jiného druhu z důvodu obrany před predátory. Cílem této práce bylo vytvořit metody na měření geometrických i fotometrických vlastností těla členovců a jejich částí z fotografií, které lze využít k vyhodnocení kvality mimeze. Aby bylo možné měřit vlastnosti členovce, musíme nejprve nalézt členovce na fotografii. Učinil jsem tak pomocí segmentace. Jedná se o úlohu spočívající v automatickém rozdělení obrazu do oblastí se stejnými vlastnostmi. Použil jsem k tomu konvoluční neuronové sítě, které jsou rychle rozvíjející se oblastí umělé inteligence a segmentační úlohu jsem řešil jako klasifikační úlohu na úrovni jednotlivých pixelů. Podařilo se mi vytrénovat neuronovou síť s úspěšností klasifikace 94.47%. Dále jsem úspěšně vyvinul metody, které s pomocí segmentovaného obrazu dokážou počítat popisné charakteristiky členovce a jeho částí související s jeho rozměry i zbarvením.

## Klíčová slova

Sémantická segmentace; konvoluční neuronové sítě; hluboké učení; U-Net; charakteristika členovců

## Annotation

Mimicry is a visual or acoustic resemblance of a different species to defend from a predator. Here I aimed to develop methods to measure geometric and photometric features of the arthropod's body and its parts in a picture, which can be used to assess mimetic quality. Firstly, the arthropod must be recognized in the image. This was done by segmentation. It is a method of an automatic division of the image into areas with similar properties. For this purpose, I used a modern approach based on convolutional neural networks and defined the segmentation as a pixel-wise classification problem. I managed to train a network with a classification accuracy of 94.47%. I also developed methods to measure the arthropod appearance features from the image based on the segmented image.

## Keywords

Semantic segmentation; convolutional neural networks; deep learning; U-Net, arthropod characterization

# **Obsah**

<b>1</b>	<b>Úvod</b>	<b>7</b>
<b>2</b>	<b>Cíle práce</b>	<b>9</b>
<b>3</b>	<b>Materiál a metodika</b>	<b>10</b>
3.1	Vstupní data . . . . .	10
3.2	Předzpracování dat . . . . .	12
3.3	Konvoluční neuronové síť . . . . .	13
3.3.1	Architektura neuronové sítě . . . . .	14
3.3.2	Přenos učení (Transfer learning) . . . . .	16
3.3.3	Chybová funkce (Loss function) . . . . .	17
3.3.4	Měření úspěšnosti . . . . .	18
3.3.5	Zpracování predikce . . . . .	19
3.4	Rozdělení na menší regiony . . . . .	20
3.5	Tvar těla . . . . .	20
3.6	Rozměry jedince . . . . .	22
3.7	Barevná charakteristika . . . . .	24
<b>4</b>	<b>Použité nástroje</b>	<b>25</b>
<b>5</b>	<b>Výsledky</b>	<b>26</b>
<b>6</b>	<b>Diskuze</b>	<b>30</b>
<b>7</b>	<b>Závěr</b>	<b>31</b>
<b>8</b>	<b>Použitá literatura</b>	<b>32</b>
<b>9</b>	<b>Seznam obrázků a tabulek</b>	<b>34</b>

# 1 Úvod

Každý živočich, aby ve volné přírodě přežil, musí najít způsob, jak se bránit před svými predátory. Některé druhy proto praktikují obrannou mimezi. Jedná se o vzhledové nebo zvukové napodobení jiného druhu (modelu), který je pro predátora jedovatý nebo nestravitelný.

Tématickou obranné mimeze se zabývají vědci z Přírodovědecké fakulty Masarykovy univerzity v Brně, kteří studují mimezi pavouků. K jejich predátorům patří savci, ptáci, plazi nebo jiní bezobratlí živočichové. Častým vzorem jsou pro ně mravenci (viz obrázek 1). Mravenci se před predátory brání zejména chemickými sloučeninami, trny na těle, žihadlem, kousnutím nebo agresivním chováním. Aby mimetický druh napodobil svůj model, musí se mu podobat velikostí, tvarem těla, barvou a pohybem [1]. Pavouci mají ale vůči mravencům odlišnou stavbu těla. Na rozdíl od mravenců mají hlavu a tělo spojené v jednu část – hlavohrud. Imitaci řeší zaškrcením nebo světlými pruhy, které simulují tři části. Dále se liší v počtu končetin. Pavouci mají čtyři páry končetin vycházející z hlavohrudi, zatímco mravenci jen tři vycházející z hrudi a k tomu na hlavě jeden páru tykadel. Proto se tito mimetici naučili používat k chůzi pouze tři páry končetin. První páru zvednou a tím napodobují tykadla.



Obrázek 1: Pavouk (vlevo) napodobující mravence (vpravo). Převzato z [https://upload.wikimedia.org/wikipedia/commons/e/ea/Ant\\_and\\_jumping\\_spider\\_Gorongosa\\_National\\_Park%2C\\_Mozambique.jpg](https://upload.wikimedia.org/wikipedia/commons/e/ea/Ant_and_jumping_spider_Gorongosa_National_Park%2C_Mozambique.jpg).

Cílem jednoho z projektů vědců z Masarykovy univerzity je zjistit, který druh mravence pavouk napodobuje. K tomu využívají bud' statické obrazy nebo videa. Hlavní výhodou statických obrazů je, že umožňují detailní popis tvaru a barevných vlastností studovaných objektů. Na druhou stranu z videa lze počítat pohybové charakteristiky [2]. Na analýzu statických obrazů byl v rámci jedné z diplomových prací na Fakultě informatiky vyvinut software [3], který měří vlastnosti pavouků a mravenců na obrázku. Základem softwaru je nalezení studovaného objektu na obrázku pomocí segmentace, což je úloha spočívající v automatickém rozdělení obrazu do oblastí se stejnými vlastnostmi. Každému pixelu obrazu je přiřazena hodnota právě jedné kategorie, kterou reprezentuje. To znamená, že se nejen snažíme určit, co se na obrázku vyskytuje, ale i přesné pozice výskytu jednotlivých pixelů.

Cílem této práce bylo využít moderních metod umělé inteligence pro zlepšení segmentace, která se ukázala být v předchozí práci málo obecná. Dále z důvodu nekompatibility implementovaných algoritmů na měření vlastností pavouků a mravenců použitých v [3] jsem vytvořil nové algoritmy, ve kterých jsem se zaměřil na popis velikosti, tvaru těla a barvy.

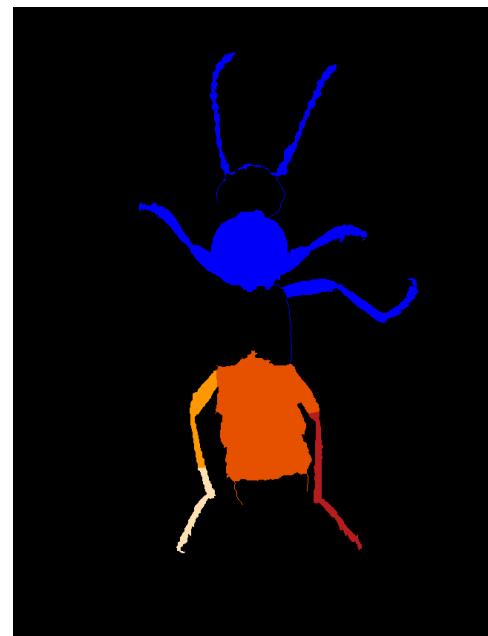
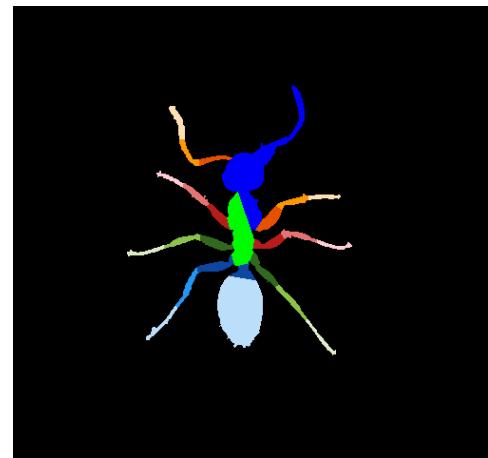
Pro segmentaci jsem použil konvoluční neuronové sítě hlubokého učení. Jedná se o rychle roz-

víjející se oblast umělé inteligence s mnoha aplikacemi v počítačovém vidění, grafice, apod. Konvoluční neuronové sítě hlubokého učení dosahují v mnoha situacích lepších výsledků než klasické metody založené na prahování, regionálních oblastech nebo detekci hran [4]. Mnou vyvinutá neuronová síť zlepšila úspěšnost segmentace oproti existující metodě ve všech měřených aspektech.

Nově vytvořené metody, popsané v této práci, jsou součástí většího softwaru na porovnávání členovců. Tento software by měl být použitelný i na klasifikaci jiných organismů. Bude volně stažitelný na stránkách Masarykovy univerzity.

## 2 Cíle práce

- Pomocí moderních metod umělé inteligence zlepšit přesnost segmentace objektu v obrazu a umožnit rozdelení na menší regiony (viz obrázek 2). Také rozšířit segmentaci z pavouků a mravenců i na jiné členovce.
- Vyvinout metody pro charakterizaci objektu zahrnující:
  - zjištění tvaru,
  - stanovení rozměrů,
  - barevnou charakteristiku.



(a) Vstupní obrázky.

(b) Masky vytvořené softwarem [3]. Barevné rozdelení je popsáno v kapitole 3.1.

Obrázek 2: Ukázka masek vytvořených předchozí prací.

## 3 Materiál a metodika

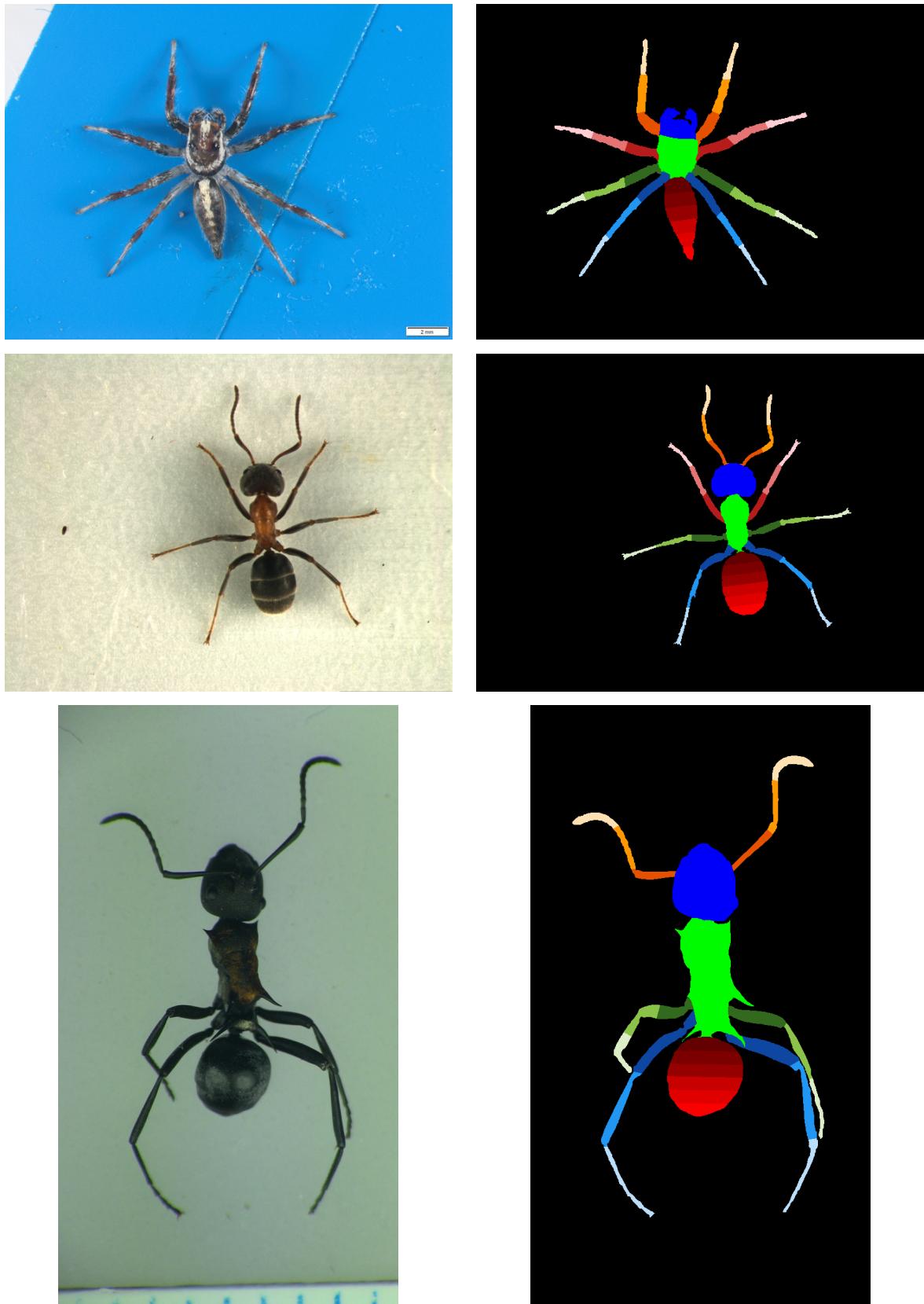
### 3.1 Vstupní data

Pro vytrénování neuronové sítě mi bylo poskytnuto pracovníky Přírodovědecké fakulty Masarykovy univerzity 1501 RGB obrázků s příslušnými šedotónovými maskami. Na každém obrázku se nacházel právě jeden objekt. Maska obsahovala různé hodnoty pixelů, které tento objekt rozdělovaly na 20 částí (viz obrázek 3b), kde každé části bylo přiřazeno unikátní číslo (na obrázcích 3b, jsou různým číslům přiřazeny různé barvy):

- 0 – pozadí (černá),
- 1 – hlava (tmavě modrá),
- 2 – hrud' (zelená),
- 3–8 – zadeček (odstíny červené, intenzita klesá od 3 do 8),
- 9–11 – první pár končetin (odstíny oranžové, intenzita klesá od 9 do 11),
- 12–14 – druhý pár končetin (odstíny červeno-oranžové, intenzita klesá od 12 do 14),
- 15–17 – třetí pár končetin (odstíny zelené, intenzita klesá od 15 do 17),
- 18–20 – čtvrtý pár končetin (odstíny modré, intenzita klesá od 18 do 20).

Zadeček byl rozdělen na 6 menších částí podél osy symetrie a končetiny rozděleny na 3 stejně dlouhé úseky v závislosti na vzdálenosti od těla. Obrázky měly tyto charakteristické vlastnosti:

- různé pozadí, nejčastěji odstín modré nebo běžové s nečistotami,
- různé velikosti,
- někdy chyběly některé končetiny,
- objekt byl natočen hlavou nahoru,
- objekt byl různobarevný, s někdy špatným kontrastem od pozadí,
- u některých obrázků se vyskytovaly výrazné odlesky,
- vyfoceny byly s různou délkou expozice.



(a) Vstupní obrázky.

(b) Referenční masky (barevně jsou odlišeny části 1 – 20, na které je objekt rozdělen).

Obrázek 3: Ukázka poskytnutých dat.

## 3.2 Předzpracování dat

Protože dodané masky neměly dostatečnou kvalitu (např. chybějící končetiny, nepřesně od-delené končetiny od ostatních částí, či špatně ořezané části), tak jsem z nich vybral jen část, konkrétně 1379 dvojic obrázek-maska, které dostatečnou kvalitu měly. Tato data jsem rozdělil na tři po dvou disjunktní množiny:

- trénovací – 1179 obrázků,
- validační – 100 obrázků,
- testovací – 100 obrázků.

Trénovací množina je určena pouze k trénování neuronové sítě (více o tréninku viz kapitola 3.3.3). Validační množina slouží k vyhodnocení vlivu hyperparametrů (parametry neuronové sítě, jejichž vliv na její úspěšnost se nedá předem určit, např. learning rate – rychlosť učení nebo batch size – počet vzorků propagovaných neuronovou sítí najednou). Vrstvy neuronové sítě jsou v trénovacím režimu, ale nemění se hodnoty vah jednotlivých vrstev, které byly nastaveny s využitím trénovací množiny. Testovací množina je určena pouze k vyhodnocení úspěšnosti neuronové sítě. Aby byly výsledky co nejprůkaznější, tak jsem do testovací množiny vybral všechny možné typy obrázků, co jsem měl k dispozici.

Jak už jsem uvedl výše, masky byly rozděleny na 20 částí, kde každá končetina byla dělena na tři stejně dlouhé díly. Toto dělení končetin může výrazně zkomplikovat trénování. Navíc některé končetiny byly v maskách upravovány ručně, což by mělo negativní vliv na přesnost rozdělení. To samé platí pro uspořádání končetin v případě, že nějaká chybí. Proto jsem zredukoval počet hledaných částí na čtyři základní: hlava, hrud', zadeček a končetiny. Referenční masku pak definujeme jako funkci  $X : \mathbb{N}_0^2 \rightarrow \{0; 1; 2; 3; 4\}$ , která každé souřadnici přiřazuje index příslušné značky. Nula reprezentuje pozadí a kladná čísla základní části, neboli objekty, dle pořadí výše. Objektem tedy rozumíme množinu pixelů patřící příslušné třídě. Rozřezání na menší regiony jsem řešil až následně, viz kapitola 3.4.

Pro zvýšení počtu trénovacích dat jsem použil datové augmentace [5]. Jedná se o metodu zvýšení počtu dat modifikovanou kopíí již existujících dat. Snažíme se tím zabránit přetrénování (stav, kdy se neuronová síť příliš optimalizuje na trénovací data a dosahuje pak špatných výsledků na datech trochu odlišných). Měla by také přispět k lepší generalizaci neuronové sítě. Využívá se v případě malého počtu originálních dat nebo minimální odlišnosti dat. V mému případě šlo o oba důvody. Proto jsem na každém obrázku z trénovací množiny před předáním na vstup neuronové sítě provedl následující náhodné geometrické transformace:

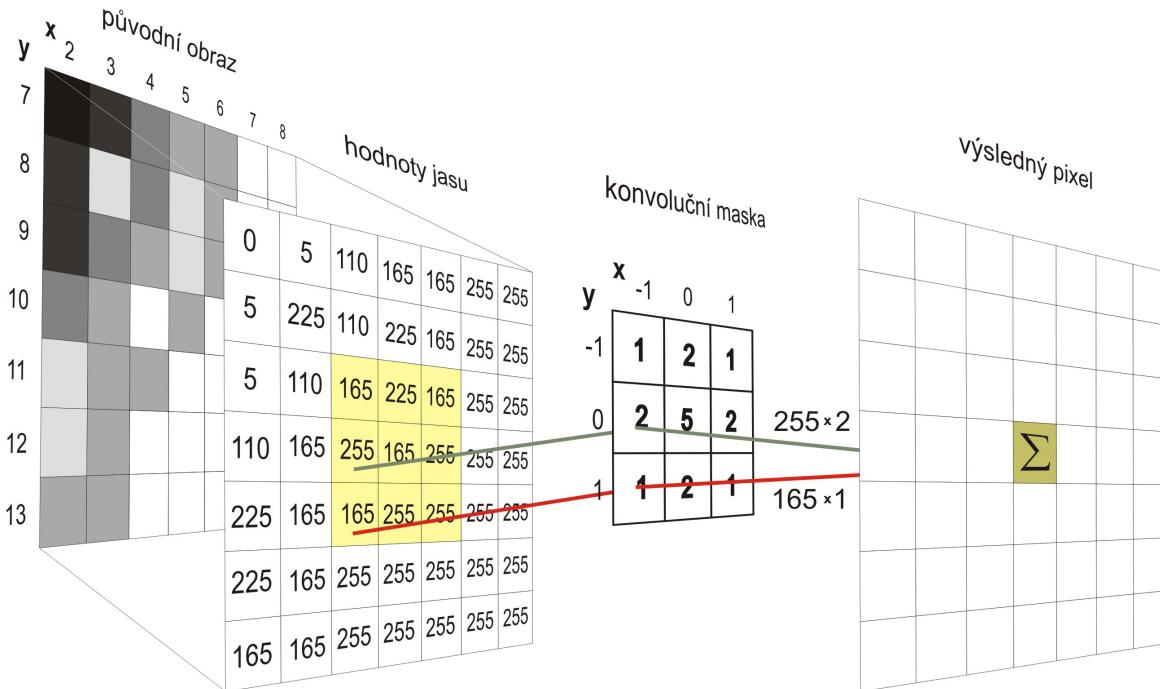
- otočení kolem středu o  $\pm 0\text{--}15^\circ$ ,
- náhodné zmenšení nebo zvětšení obrázku v rozsahu 0–10%,

- horizontální překlopení,
- náhodné posunutí na obou osách v rozsahu 0–10% výšky/šířky obrazu.

Při některých transformacích je třeba doplnit informaci v obrázku mimo oblast jeho definičního oboru. To jsem učinil vložením původního obrázku na nedefinovaná místa. Vzniká tím velmi důvěryhodné pozadí, které většinou i dokonale navazuje na pozadí transformovaného obrázku. Tato metoda může způsobit to, že se nějaká část objeví dvakrát. To nám ale nevadí, protože na některých původních obrázcích se vyskytují odlomené končetiny, které už do masky nechceme započítat. Tímto postupem tak pouze vytvoříme další obrázky obsahující komplikované pozadí, které nechceme segmentovat.

### 3.3 Konvoluční neuronové sítě

Konvoluční neuronová síť je neuronová síť obsahující konvoluční vrstvy. Konvoluční vrstva aplikuje na vstupní obraz konvoluci o zvolené velikosti. Konvoluční jádro obsahuje hodnoty (váhy), kterými jsou jednotlivé pixely vynásobeny. Součtem vynásobených pixelů získáme skalar, který tvoří pixel příznakové mapy (feature map). Konvoluce je pozičně nezávislá a stejné konvoluční jádro je aplikováno na všechny pixely, proto na počtu a pozici pixelů nezáleží. Celý postup je názorně zobrazen na obrázku 4.



Obrázek 4: Princip fungování konvoluční vrstvy. Převzato z [https://commons.wikimedia.org/wiki/File:Konvoluce\\_2rozm\\_diskretni.jpg](https://commons.wikimedia.org/wiki/File:Konvoluce_2rozm_diskretni.jpg).

Konvoluční jádro se posouvá o  $S$  kroků po celém obrázku. Nejčastěji o jeden krok. V případě, kdy  $S > 1$  a jádro nedojde až k okraji obrázku, tak se chybějící hodnoty nahradí nulami (tzv.

zero-padding). Vzniká tím černý rámeček kolem obrázku. Velikost nového obrazu spočítáme jako:

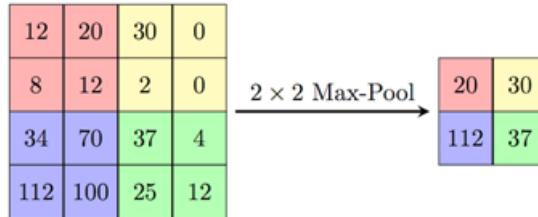
$$N = \frac{W - F + 2P}{S} + 1,$$

kde  $W$  je šířka/výška vstupního obrazu,  $F$  velikost konvolučního jádra a  $P$  počet sloupců/řádků, které se doplňují nulami. Konvolucí dojde k zmenšení původního obrazu. V případě použití  $S = 1$  chceme většinou zachovat stejnou velikost jako u vstupního obrazu. Velikost zero-padding je pak rovna:

$$P = \frac{F - 1}{2}.$$

Pro zvýšení modelovacích schopností neuronových sítí a zlepšení učení pro složitá data se používají tzv. aktivační funkce. Aktivační funkce jsou nelineární funkce, které výstup jednotlivých neuronů transformují dle svého typu.

Další významná vrstva konvolučních neuronových sítí je sdružení (pooling). Sdružení progresivně redukuje hustotu dat. Jedná se o masku určité velikosti, která vybírá ze svého rozsahu pouze jeden pixel určité vlastnosti. Tím dochází k redukci dat, což urychluje učení a dává možnost následným konvolucím náhled na větší část obrazu.



Obrázek 5: Vlevo vstupní obraz, vpravo po aplikaci  $2 \times 2$  max-pooling. Převzato z [https://embarc.org/embarc\\_mli/doc/build/html/MLI\\_kernels/pooling\\_max.html](https://embarc.org/embarc_mli/doc/build/html/MLI_kernels/pooling_max.html)

Nejpoužívanější je max-pooling (viz obrázek 5). Jak z názvu vyplývá, vybere pouze pixely s nejvyšší hodnotou. Kvůli předchozím konvolucím, které násobí jednotlivé pixely určitými váhami, jsou ty s nejvyšší hodnotou považovány za nejdůležitější. Tím extrahujeme z obrazu pouze ty nejdůležitější rysy.

### 3.3.1 Architektura neuronové sítě

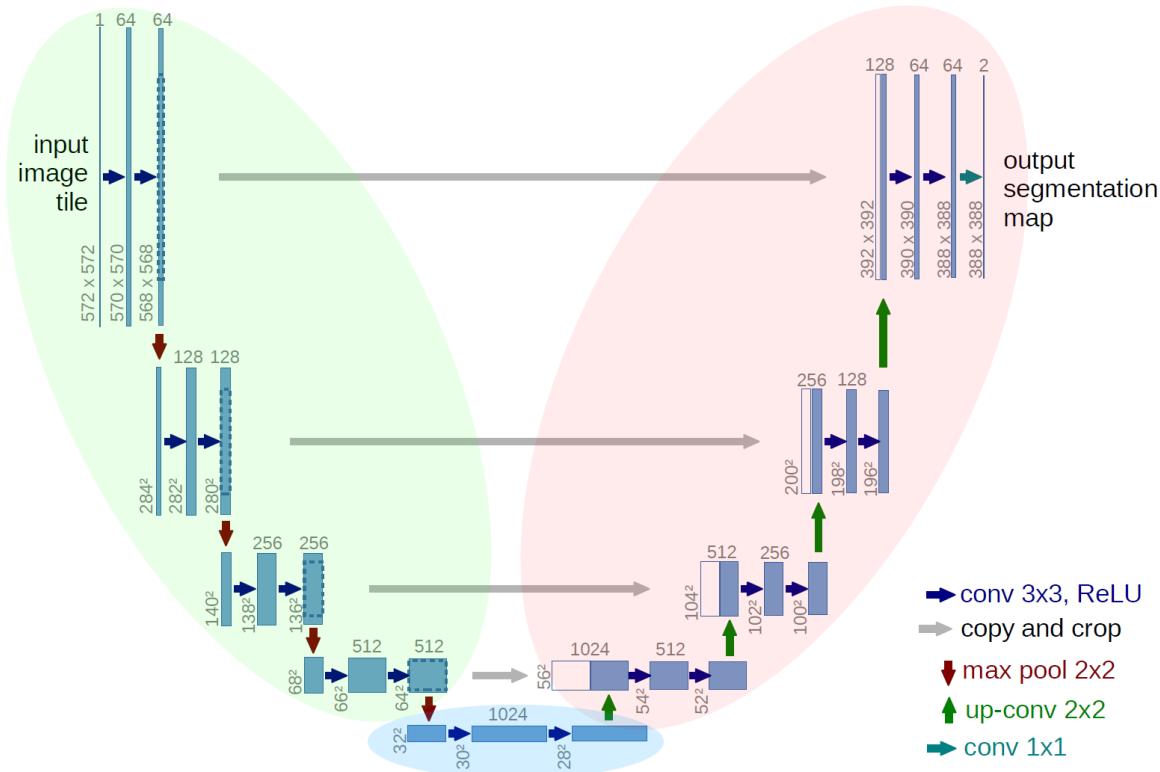
Pro segmentaci v této práci jsem použil architekturu zvanou U-Net [6]. Byla vytvořena pro segmentaci biomedicínského obrazu s malou trénovací množinou. Jedná se o plně konvoluční síť (Fully Convolutional Networks). To znamená, že obsahuje pouze konvoluční vrstvy, a proto ji lze jednoduše použít pro vstupní obrazy o různé velikosti.

U-Net architekturu rozdělujeme na tři části (viz obrázek 6):

- smršťovací část (contracting path) – extrahuje důležitých informace (rysy),
- oblast zúžení (bottleneck) – propouští pouze nejdůležitější rysy,
- expanzní oblast (expansive path) – lokalizace extrahovaných rysů a vytvoření výstupu,

kde jednotlivé bloky se skládají z těchto částí:

- konvoluční vrstva (velikost konvolučního jádra  $3 \times 3$ ),
- aktivační funkce
- normalizace instance (Instance normalization)
- ředění (Dropout),
- Max-pooling (velikost masky  $2 \times 2$ )



Obrázek 6: Architektura konvoluční neuronové sítě U-Net. Zeleně označena smršťovací část, modře oblast zúžení a růžově expanzní oblast. Převzato z <https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/u-net-architecture.png>.

Výstup konvoluční vrstvy ve vnitřních vrstvách neuronové sítě transformuje aktivační funkce ReLu (Rectified Linear Unit), která nastaví všechny záporné hodnoty na nulu a kladné nemění. Je výpočetně velice efektivní, díky tomu je vhodná pro hluboké neuronové sítě [7]. Byla také použita autory U-Net. Na výstupní vrstvě jsem k vytvoření predikce aplikoval sigmoidní funkci

(logistická funkce). Dosahuje hodnot 0–1, což je v méém případě, kdy je na jednotlivém kanále pouze pozadí (0) a hledaná část (1), ideální.

V neuronových sítích hlubokého učení často dochází k přetrénování. Využívá se proto technika ředění [8]. Jedná se o metodu, která náhodně ignoruje námi zvolené procento neuronů z předchozí vrstvy při trénování. Snažíme se tím zabránit přetrénování. Ředění přispívá také ke generalizaci sítě. Ignorováním náhodných neuronů bráníme síti si zapamatovat jednotlivé trénovací instance a nutíme ji vytvořit obecnější model vstupních dat. Ve vyhodnocovacím režimu, se žádné neurony neignorují, a proto je třeba výstupy neuronů dané vrstvy snížit podle procenta ignorovaných neuronů při tréninku, aby nedocházelo k příliš vysoké odezvě vrstvy. Já jsem ředění použil ve vnitřních blocích U-Net po konvoluční vrstvě.

Přidal jsem navíc normalizaci instance [9]. Jedná se o techniku používanou k normalizaci vstupních vektorů uvnitř sítě. Použil jsem ji po konvolučních vrstvách ve smršťovací části. Normalizuje tím hodnoty transformované aktivační funkcí. Zrychluje tím učení neuronové sítě. Snaží se řešit to, aby neuronová síť nebyla závislá na kontrastu. Také pomáhá generalizaci a brání přetrénování, takže dokáže zmenšit velikost ředění, čímž snižuje ztrátu dat.

### 3.3.2 Přenos učení (Transfer learning)

Trénování celé neuronové sítě je velice náročný proces a je k němu potřeba spousta dat. Existuje už však množství neuronových sítí vytrénovaných na miliónech dat. Proto se v praxi používají tyto předučené (pre-trained) sítě, které se doladí do námi potřebné podoby přidáním nebo odebráním některých vrstev. Výhodou je rychlejší trénování, větší obecnost a někdy i lepší úspěšnost. Nevýhodou může být velikost této sítě, protože se většinou jedná o obrovské sítě, ke kterým ještě přidáme další vrstvy.

V této práci jsem použil dvě předučené sítě: VGG-16 [10] a ResNet34 [11]. Byly vytrénované na ImageNet databázi pro soutěž ILSVRC (ImageNet Large Scale Visual Recognition Challenge) [12], s 1.3 mil trénovacích obrázků v 1000 kategoriích. Použil jsem je jako enkodér (neuronová síť určená k extrakci důležitých rysů ze vstupních dat) v U-Net architektuře místo smršťovací části.

VGG-16 je hluboká konvoluční neuronová síť hlubokého učení. Na ILSVRC dosáhla v top-5 úspěšnosti 90.1%. Obsahuje 16 konvolučních vrstev. Dá se velmi lehce propojit s U-Net architekturou, ale je hodně pomalá při predikcích a také při trénování, jelikož obsahuje přibližně 138 miliónů parametrů.

ResNet34 je oproti VGG-16 ještě hlubší. Obsahuje 34 konvolučních vrstev, které ale pracují s mnohem menším počtem kanálů (moto: „thinner but deeper“), díky čemuž je mnohem rychlejší (až 5×) než VGG-16. Má také mnohem méně parametrů okolo (25.6 miliónů), čímž se rychleji trénuje. Na ILSVRC dosáhla v top-5 úspěšnosti 92.1%.

Při učení obou enkodérů se tzv. zmrazí jejich vrstvy, aby se hodnota jejich parametrů neměnila.

Trénují se pouze námi přidané vrstvy. Po vytrénování můžeme použít metodu fine-tuning, kdy se vrstvy enkodéru odmrazí a trénují se s malým learning rate (rychlosť učení).

### 3.3.3 Chybová funkce (Loss function)

K trénování konvolučních neuronových sítí se nejčastěji používá metoda optimalizace stochastického gradientního sestupu [13]. Hodnota vah se optimalizuje pomocí zpětného šíření chyby (back-propagation). Chybová funkce slouží k vypočítání této chyby. Trénováním se tedy neuronová síť snaží minimalizovat chybu, která je počítána jako chyba mezi predikcí a referenční segmentací. Objekt referenční segmentace jsem definoval jako funkci  $r_c$ :

$$r_c : \mathbb{N}_0^2 \rightarrow \{0; 1\},$$

$$r_c(i) = \begin{cases} 1, & X(i) = c, \\ 0, & \text{jinak,} \end{cases}$$

a objekt predikovaný neuronovou sítí jako funkci  $p_c$ :

$$p_c : \mathbb{N}_0^2 \rightarrow \langle 0; 1 \rangle,$$

$$p_c(i) = Y(i)_c,$$

kde funkce  $Y : \mathbb{N}_0^2 \rightarrow \langle 0; 1 \rangle^4$  je maska predikovaná neuronovou sítí. Jako chybovou funkci jsem použil tzv. Dice Loss. Je založená na Sørensen–Dice koeficientu [14]. Jedná se o regionálně zaměřenou chybovou funkci. Nebere v potaz správně určené pozadí, které je ve vstupních obrazech mnohem častější než pixely patřící objektu. Pro objekt  $c$  je definována jako:

$$DL_c(r_c, p_c) = \frac{2 \sum_{i \in \mathbb{N}_0^2} r_c(i)p_c(i)}{\sum_{i \in \mathbb{N}_0^2} r_c(i) + \sum_{i \in \mathbb{N}_0^2} p_c(i)}.$$

Kvůli nerovnoměrnému zastoupení objektů, jsem přidal adaptivní váhy, které dávají váhu  $v_c$  jednotlivým objektům dle chyby vůči objektu s nejnižší chybou (čím větší chyba, tím větší váha):

$$v_c(r_c, p_c) = \frac{\max_c \{DL_c(r_c, p_c)\}}{DL_c(r_c, p_c)}.$$

Váženou Dice Loss  $DL$  pak spočítáme:

$$DL(X, Y) = 1 - \sum_{c=1}^{|C|} \frac{v_c(r_c, p_c) DL_c(r_c, p_c)}{\sum_{c \in C} v_c(r_c, p_c)},$$

kde  $C$  je množina objektů. Objekty masky by měly tvořit jeden celek. Dice Loss ale počítá chybu na každém objektu zvlášť, čímž nemá žádný vliv na výslednou celistvost objektů. Navíc

je hlavohrud' u pavouků v referenčních datech rozdělena na dvě části, ale reálně se jedná jen o jednu. Tím pádem neexistuje přesná hranice rozdělení. To by mohlo způsobit, že objekty v predikovaných maskách nebudou tvořit jeden celek, což by mělo negativní dopad na následné rozdělení do menších regionů. Přidal jsem proto druhou chybovou funkci  $TL$  (1), založenou na Tverskyho indexu [15]. Tato funkce počítá chybu na celém objektu a ne na jednotlivých objektech.

$$\begin{aligned} TP(r_m, p_m) &= \sum_{i \in \mathbb{N}_0^2} [r_m(i)p_m(i)], \\ FP(r_m, p_m) &= \sum_{i \in \mathbb{N}_0^2} \{p_m(i)[1 - r_m(i)]\}, \\ FN(r_m, p_m) &= \sum_{i \in \mathbb{N}_0^2} \{r_m(i)[1 - p_m(i)]\}, \\ TL &= 1 - \frac{TP + \gamma}{TP + \alpha FP + \beta FN + \gamma}, \end{aligned} \quad (1)$$

$$\begin{aligned} \text{kde } r_m : \mathbb{N}_0^2 &\rightarrow \{0; 1\}, \\ r_m(i) &= \begin{cases} 1, & X(i) > 0, \\ 0, & \text{jinak}, \end{cases} \\ \text{a } p_m : \mathbb{N}_0^2 &\rightarrow \langle 0; 1 \rangle, \\ p_m(i) &= \max_c(p_c(i)). \end{aligned}$$

Hyperparametr  $\alpha$  je váha špatně určených pixelů a  $\beta$  váha odchylky od správně určeného pixelu.  $\alpha + \beta = 1 \Rightarrow \beta = 1 - \alpha$ . Pokud  $\alpha = \beta$ , pak se jendá o Dice Loss.  $\gamma$  je vyhlazovací koeficient. Výsledná chybová funkce  $L$  je váhový průměr obou funkcí:

$$L(w) = wDL + (1 - w)TL,$$

kde  $w$  je hyperparametr. Cílem trénování bylo minimalizovat hodnotu  $L$ . Při počítání chyby na validační množině je výsledná chybová funkce aritmetický průměr obou funkcí. Hodnoty hyperparametrů jsou určeny v kapitole 5.

### 3.3.4 Měření úspěšnosti

K určení úspěšnosti neuronové sítě jsem použil tři funkce. Úspěšnost byla vyhodnocena aritmetickým průměrem na celé testovací množině. Jelikož poslední konvoluční vrstva v U-Net je následována sigmoid aktivační funkcí, tak úspěšnost dosahuje hodnot 0 – 1.

Pro určení celkové úspěšnosti segmentace jsem použil metodu založenou na průměrování přes

všechny správně určené pixely (Mean Pixel-wise),  $P$ :

$$P(X, Y) = \frac{\sum_{c=1}^{|C|} P_c(r_c, p_c)}{|C|},$$

kde  $P_c$  se spočítá následovně:

$$P_c(r_c, p_c) = \frac{\sum_{i \in \mathbb{N}_0^2} [1 - |r_c(i) - \lfloor p_c(i) \rfloor|]}{|R|},$$

kde  $\lfloor p_c(i) \rfloor$  je zaokrouhlená hodnota predikce a  $R$  je množina pixelů objektu. Pro vyhodnocení úspěšnosti rozdělení na části jsem použil Mean IoU,  $IoU$  (2), neboli Jaccardův koeficient [16]. Úspěšnost je počítána jako podíl průniku a sjednocení mezi referenční segmentací a predikcí. Zanedbává správně určené pozadí, které, jak už jsem zmínil, je ve vstupních obrazech mnohem častější než pixely patřící objektu.

$$IoU(X, Y) = \frac{\sum_{c=1}^{|C|} IoU_c(r_c, p_c)}{|C|}, \quad (2)$$

kde  $IoU_c$  spočítáme jako:

$$IoU_c(r_c, p_c) = \frac{\sum_{i \in \mathbb{N}_0^2} [r_c(i) \lfloor p_c(i) \rfloor]}{\sum_{i \in \mathbb{N}_0^2} [r_c(i) + \lfloor p_c(i) \rfloor] - \sum_{i \in \mathbb{N}_0^2} [r_c(i) \lfloor p_c(i) \rfloor]}.$$

Pro zjištění úspěšnosti segmentace celého objektu jako celku jsem použil Binary IoU,  $BIoU$ :

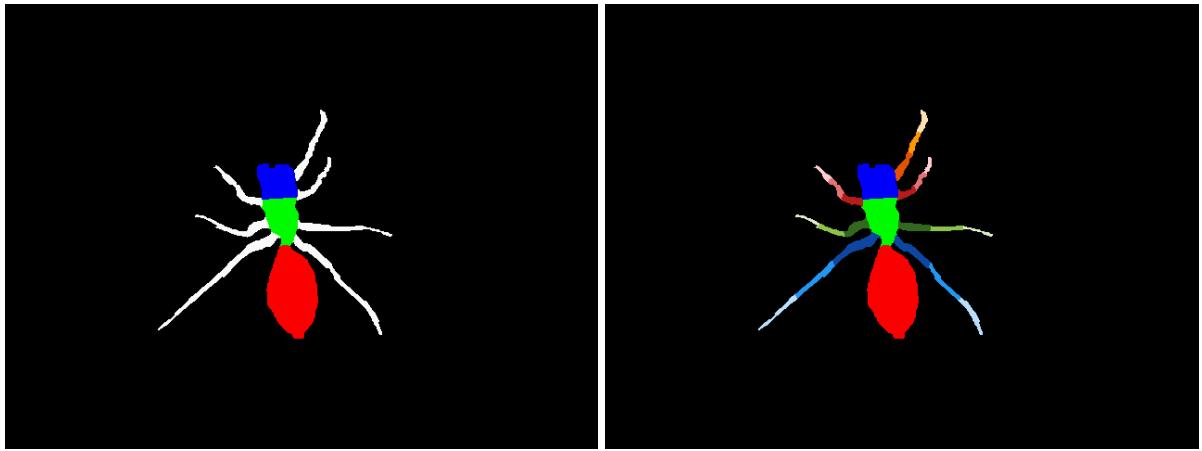
$$BIoU(r_m, p_m) = \frac{\sum_{i \in \mathbb{N}_0^2} [r_m(i) \lfloor p_m(i) \rfloor]}{\sum_{i \in \mathbb{N}_0^2} [r_m(i) + \lfloor p_m(i) \rfloor] - \sum_{i \in \mathbb{N}_0^2} [r_m(i) \lfloor p_m(i) \rfloor]}.$$

### 3.3.5 Zpracování predikce

Jelikož poslední vrstva U-Net obsahuje sigmoidní aktivační funkci, tak nejprve na predikci neuronové sítě aplikujeme práh o velikosti 0.5. Neuronová síť klasifikuje pixely do čtyř tříd: hlava, hrud', zadeček a končetiny. Predikce neuronové sítě může obsahovat více instancí hledaných objektů než hledáme. Proto nejprve převedeme predikovanou masku do binárního obrazu, kde hodnotu jedna mají pixely patřící alespoň jednomu objektu. V binárním obrazu označíme pixely algoritmem pro hledání souvislých komponent a ponecháme pouze tu s největší plochou a ty, jejichž plocha je alespoň 1/20 plochy největší komponenty a zároveň jejichž euklidovská vzdálenost od největší komponenty je menší než 20 (mohlo by se jednat o nespojený objekt). Poté znova aplikujeme hledání souvislých komponent, ale tentokrát na každém objektu zvlášť. Necháme pouze tu největší (v případě nohou osm největších) a zbylým přiřadíme index nejbližšího objektu.

### 3.4 Rozdělení na menší regiony

Objekty pro účely dalšího zpracovávání dále dělíme na několik menších částí. Končetiny rozdělujeme na čtyři páry. Učinil jsem tak podle jejich vertikální pozice v obrázku. V případě, že nějaká končetina chybí, jsou končetiny uspořádány podle osy symetrie (viz níže). Každá končetina je dále rozdělena na tři stejně dlouhé části. Pro dělení jsem využil Dijkstruv algoritmus pro hledání nejkratší cesty [17]. Pixelům sousedícím s tělem jsem dal vzdálenost nula a konečný pixel  $P_l$  byl ten s největší tzv. geodetickou vzdáleností od iniciálního bodu. Geodetická vzdálenost dvou bodů je délka nejkratší cesty, která prochází pouze přes povolené pixely (v tomto případě pixely příslušné končetiny). Vzdálenost bodu  $P_l$  nazýváme délkou končetiny. Dělení provádíme tak, že do každé části přiřadíme pixely na základě jejich vzdálenosti od pixelů sousedících s tělem. V první části mají všechny pixely geodetickou vzdálenost menší než  $1/3$  délky končetiny. V poslední části mají všechny pixely vzdálenost větší než  $2/3$  délky končetiny (viz obrázek 7).



(a) Predikce neuronové sítě.

(b) Spárované končetiny rozdělené na tři stejně dlouhé části.

Obrázek 7: Ukázka rozdělení končetin na menší regiony.

Zadeček byl původně rozdělen na šest menších částí. Po diskuzi s vědci z Masarykovy univerzity jsme se dohodli, že jej dělit nebudou potřebovat. Pokud by to potřeba bylo, tak by bylo možné použít podobný algoritmus jako pro dělení nohou.

### 3.5 Tvar těla

K určení tvaru těla, tj. nalezení pixelů na okraji příslušných souvislých komponent (hlava, hrud' a zadeček) musíme nejprve najít osu symetrie těla. Jelikož se členovci mohou ohýbat ve spojích (viz obrázek 8), tak hledáme osu symetrie každého objektu (hlava, hrud', zadeček) zvlášt'. Ostu jsem reprezentoval ve směrnicovém tvaru přímky  $x = ky + c$ , kde  $x, y$  jsou souřadnice pixelu,  $k$  je směrnice osy a  $c$  je posunutí na ose  $x$ . Prohodil jsem souřadnice, protože přímka nemůže být rovnoběžná s osou  $y$  (v mém případě  $x$ ).

Pro nalezení osy každé části jsem si nejprve našel těžiště, které jsem považoval za střed objektu



Obrázek 8: Červeně jsou vyznačené spoje, ve kterých se může mravenec ohýbat.

a osa jím musela procházet. Souřadnice těžiště  $\bar{x}, \bar{y}$  jsem spočítal jako:

$$\bar{x} = \frac{1}{|F|} \sum_{x \in F} x,$$

$$\bar{y} = \frac{1}{|F|} \sum_{y \in F} y,$$

kde  $F$  je množina pixelů dané části. Poté jsem si našel množinu souřadnic středových bodů  $M$ :

$$M = \left\{ (x, y) \mid \frac{x_r + x_l}{2}, y \in R \right\},$$

kde  $x_r$  je pravý okraj na řádku  $y$  objektu a  $x_l$  levý.  $R$  je množina  $y$  souřadnic objektu. Směrnici natočení osy symetrie  $k$  jsem spočítal metodou nejmenších čtverců:

$$k = \frac{\sum_{(x,y) \in M} (x - \bar{x})(y - \bar{y})}{\sum_{(x,y) \in M} (y - \bar{y})^2},$$

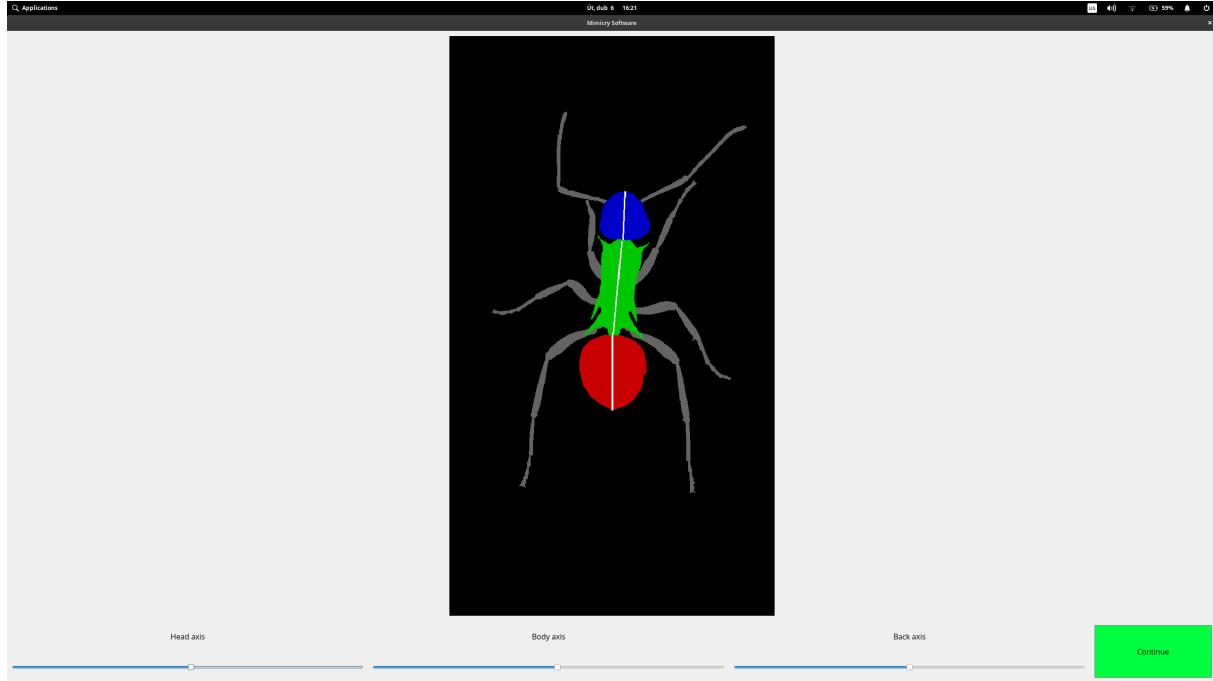
a  $c$  je pak:

$$c = \bar{x} - k\bar{y}.$$

Na těle se vyskytují výrůstky, které silně mohou ovlivnit natočení osy. Proto jsem přidal ještě jeden krok. Spočítal jsem si medián  $m$  absolutních odchylek  $M$  od osy:

$$m = \text{med} (\{|ky + c - x|; (x, y) \in M\}).$$

Pak znovu proběhla metoda nejmenších čtverců, ale tentokrát jen s body, jejichž absolutní odchylka je menší nebo rovna  $m$ . Pro dosažení co největší přesnosti je možné manuálně upravit natočení os symetrie v uživatelském prostředí (viz obrázek 9).



Obrázek 9: Grafické uživatelské prostředí pro úpravu os symetrie.

Po nalezení os symetrie jsem zprůměroval vzdálenost levé a pravé části každé části s využitím normálového vektoru osy, abych získal tzv. tvar těla. V místech, kde se normálové vektory jedné osy protínají s normálovými vektory druhé, je výsledná vzdálenost ta delší.

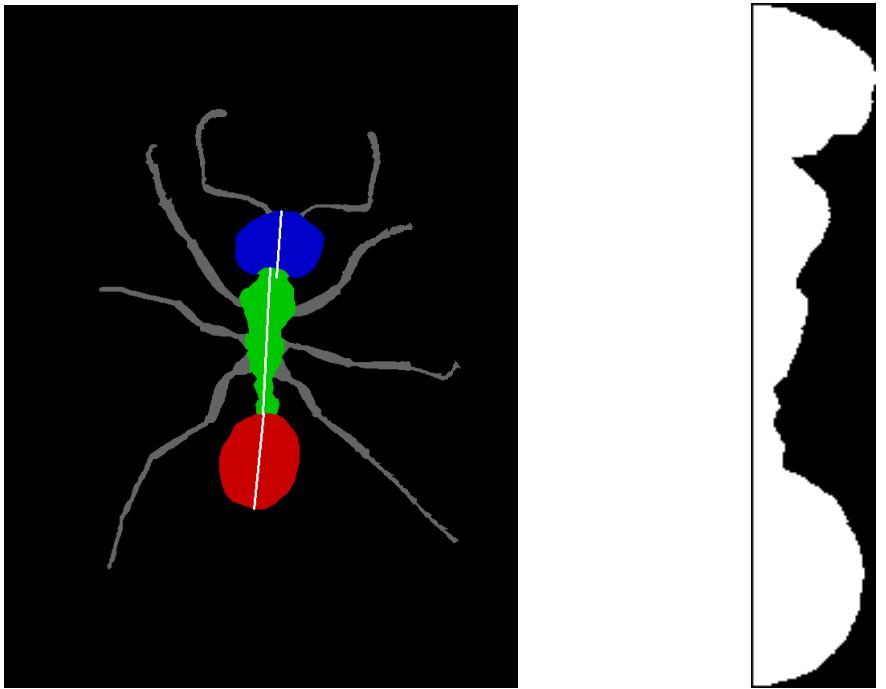
Aby se dal porovnat tvar těla mezi jednotlivými členovci, musí být délka těla stejná u všech obrázků. Proto jsou normalizovány na délku 300. V případě potřeby jsou chybějící hodnoty dopočítány pomocí lineární interpolace. Ukázka výstupu celého algoritmu je na obrázku 10.

## 3.6 Rozměry jedince

Rozměr jedince je vyjádřen délkou těla a šírkou končetin. Délku těla spočítáme jako:

$$D = \frac{1}{\alpha} \sum_{p \in P} p,$$

kde  $\alpha$  je kolik pixelů na obrázku reprezentuje 1 cm ve skutečnosti. Tento parametr uživatel zadá v aplikaci.  $P$  je množina délek jednotlivých objektů těla. Délku objektu jsem počítal dvěma způsoby. První metoda spočítá délku jako vzdálenost mezi krajními body osy symetrie (viz obrázek 11a). Tím pádem nebene v potaz kusadla pavouků nacházející se na hlavohrudi a také



(a) Osa symetrie mravence.

(b) Graf tvaru těla.

Obrázek 10: Ukázka osy symetrie a tvaru těla.

jiné výrůstky na těle. Druhá metoda počítá délku jako vzdálenost mezi krajními kolmicemi na osu symetrie (viz obrázek 11b). Bere tím v potaz všechny výrůstky.

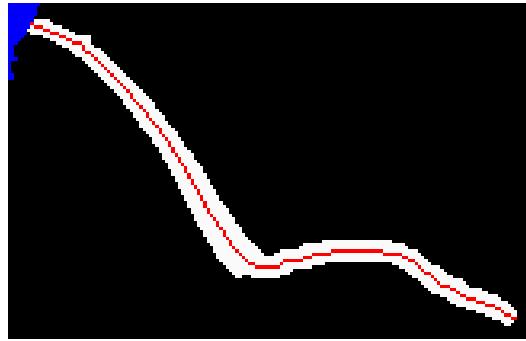


Obrázek 11: Metody měření délky hlavové části.

Tloušťku části končetiny,  $T$ , počítáme jako:

$$T = \frac{3|F|}{S},$$

kde  $F$  je množina pixelů dané části končetiny a  $S$  je odhad délky končetiny, kterou spočítáme pomocí kostry končetiny (viz obrázek 12). Pro nalezení kostry končetiny jsem použil algoritmus popsaný v [18].



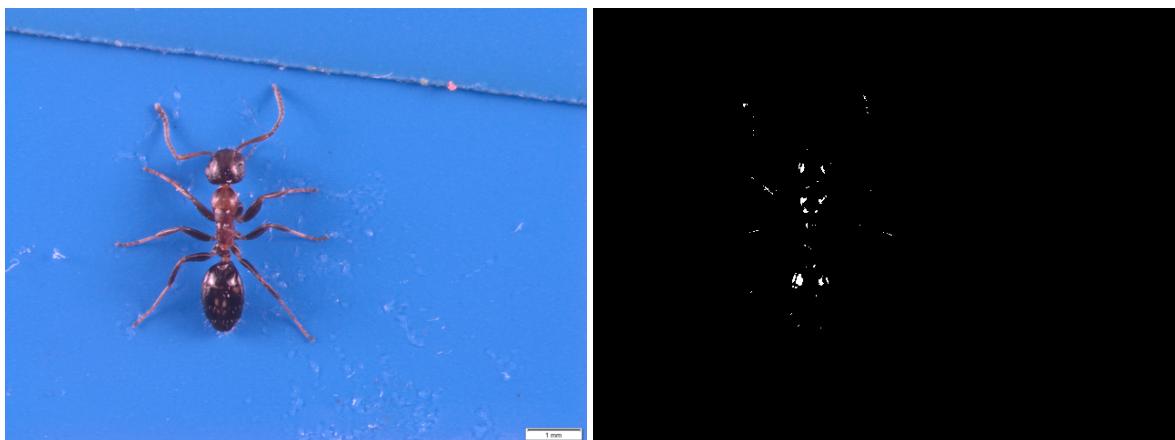
Obrázek 12: Červená čára vyznačuje kostru končetiny.

### 3.7 Barevná charakteristika

Aby mimetický druh napodobil svůj model musí se mu podobat zbarvením. Zbarvení jsem vyjádřil histogramem objektu pro každou barevnou složku (RGB) vstupního obrazu zvlášť. Pro zjištění kvality napodobení pak porovnáme jejich histogramy. Obrázky obsahují spoustu odlesků způsobených osvětlením mikroskopu. Tyto odlesky mohou výrazně ovlivnit kvalitu tohoto histogramu. Proto jsem si nejprve vytvořil masku odlesků. Obrázek jsem převedl do HSL barevného spektra. Dále už jsem počítal jenom s L kanálem, který jsem definoval jako funkci  $L : \mathbb{N}_0^2 \rightarrow \{0 \dots 255\}$ . Spočítal jsem průměrnou hodnotu pixelu objektu  $\bar{p}$  s využitím segmentované masky. Končetiny jsem bral jako jeden objekt z důvodu malé plochy, čímž by byla při výskytu odlesku silně ovlivněna průměrná hodnota. Spočítal jsem směrodatnou odchylku  $\sigma$  pixelů v objektu:

$$\sigma = \sqrt{\frac{1}{|P|} \left( \sum_{p \in P} L(p) - \bar{p} \right)^2},$$

kde  $P$  je množina pixelů objektu na kanále L. Masku  $M$  odlesků jsem definoval jako množinu pixelů, jejichž hodnota v L byla menší než  $\bar{p} + 2\sigma$  (viz obrázek 13). Při počítání histogramů se použijí pouze ty pixely vstupního obrazu, na kterých se odlesk na  $M$  nevyskytuje. Aby bylo možné histogramy porovnávat, jsou jejich hodnoty normalizovány na škálu 0 – 1.



(a) Vstupní obrázek.

(b) Maska odlesků.

Obrázek 13: Ukázka masky odlesků.

## 4 Použité nástroje

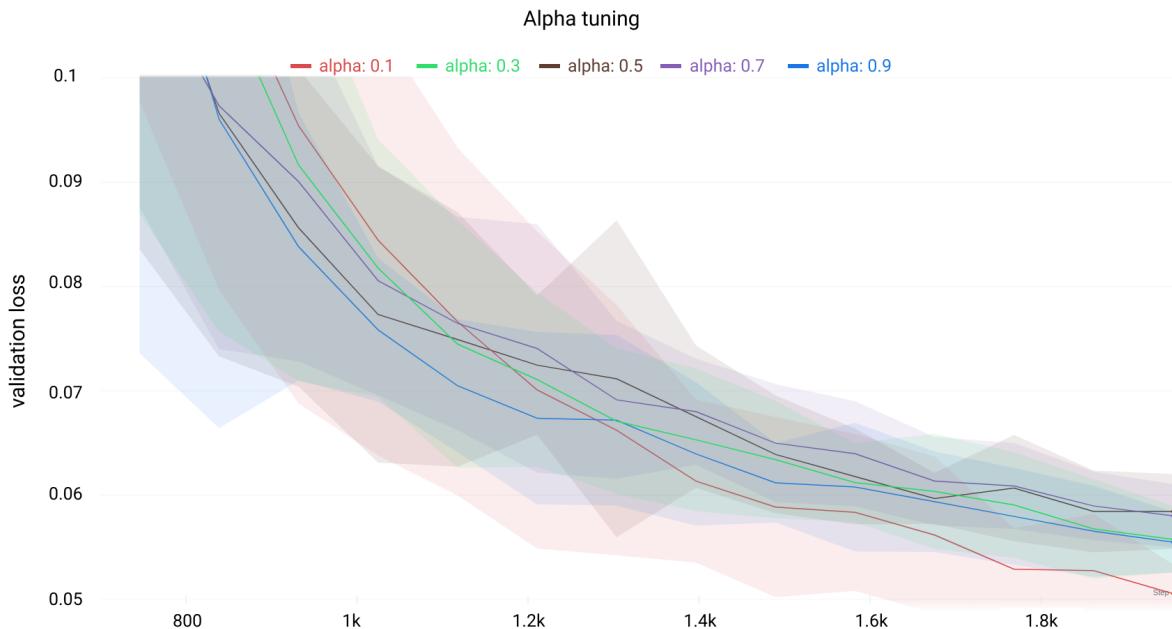
Neuronové sítě jsem vytvořil a vytrénoval v jazyce Python. Pro práci s obrázky jsem použil knihovnu OpenCV [19], která nabízí rozšířenou práci s videozáznamy a obrázky. Na tvorbu neuronových sítí a pro rychlé počítání s vektory jsem použil knihovnu PyTorch [20]. Spuštění neuronové sítě je napsáno v jazyku C++ kvůli vyšší výkonosti. Výsledky jsem vizualizoval pomocí Weights & Biases [21]. Neuronové sítě jsem trénoval na GPU Nvidia Quadro RTX 6000 na Linode cloud a GeForce RTX 2070 SUPER poskytnuté mým mentorem.

Všechny ostatní metody jsem vytvořil v jazyku C++. Pro tvorbu grafického prostředí jsem použil knihovnu Qt [22]. Metody byly vytvořeny pro operační systém Windows 10, ale je možné zkompilovat pro použití v systémech Linux a Mac OS.

## 5 Výsledky

Pro splnění hlavního cíle práce tj. zlepšit hledání, rozdelení hledaného objektu na menší regiony a také rozšíření segmentace z pavouků a mravenců i na jiné členovce, jsem se zaměřil na detailní vyhodnocení segmentační části založené na neuronových sítích. Pro vytvoření co nejúspěšnější neuronové sítě, je nutné najít správnou kombinaci hyperparametrů. V mém případě to byly:  $\alpha$  u  $TL$ , váha  $w$  chybových funkcí, *enkovér* U-Net architektury, *batch size* a *learning rate*. Vytrénování sítí se všemi možnými kombinacemi hyperparametrů je výpočetně velice náročné, proto jsem hledání rozdělil na dvě části. Nejprve jsem hledal hodnoty  $\alpha$  a  $w$ , jelikož by neměly být přímo závislé na architektuře sítě. Použil jsem k tomu samostatnou U-Net architekturu, protože je rychlejší než s použitím předučeného enkovéru. Trénink proběhl ve dvaceti epochách (iterace na celé trénovací množinou). Validační chyba vyla vyhodnocována po každé epoše na celé validační množině. Podařilo si mi vytrénovat 161 neuronových sítí, kdy hodnoty hyperparametrů byly vzájemně testovány každý s každým. Pro vybrání těch nejúspěšnějších jsem zvolil validační chybu menší než 0.065. Tímto zbylo 132 neuronových sítí.

$\alpha$  jsem vybíral z hodnot: 0.1, 0.3, 0.5, 0.7, 0.9 podle článku [23]. Validační chybu jsem zprůměroval podle  $\alpha$  hodnoty. Validační chyba dosahovala nejnižších hodnot v případě  $\alpha = 0.1$  (viz obrázek 14). Validační úspěšnost  $IoU_m$  naopak dosahovala nejvyšších hodnot pokud  $\alpha = 0.3$  a nejnižších v případě  $\alpha = 0.1$  (viz obrázek 15). Důvod je ten, že validační chyba se počítá s danou hodnotou  $\alpha$ , čímž není objektivní. Proto za nejúspěšnější považujeme  $\alpha = 0.3$ .

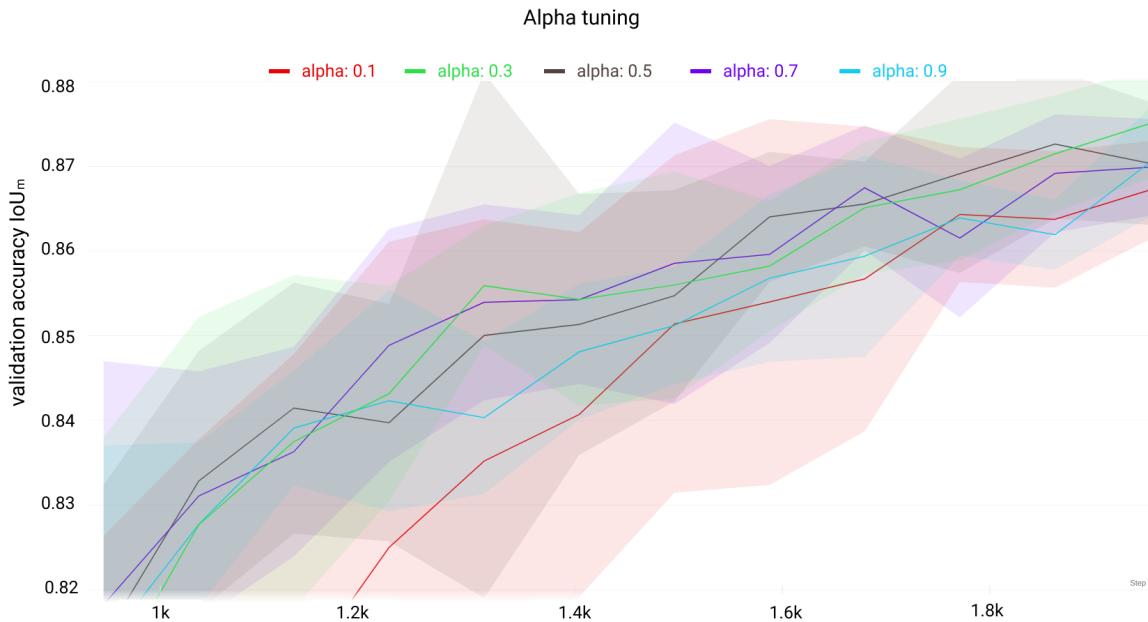


Obrázek 14: Průměrná validační chyba neuronových sítí s různými hodnotami  $\alpha$ . Světlou barvou je vynáčena směrodatná odchylka pro hodnoty  $\alpha$ .

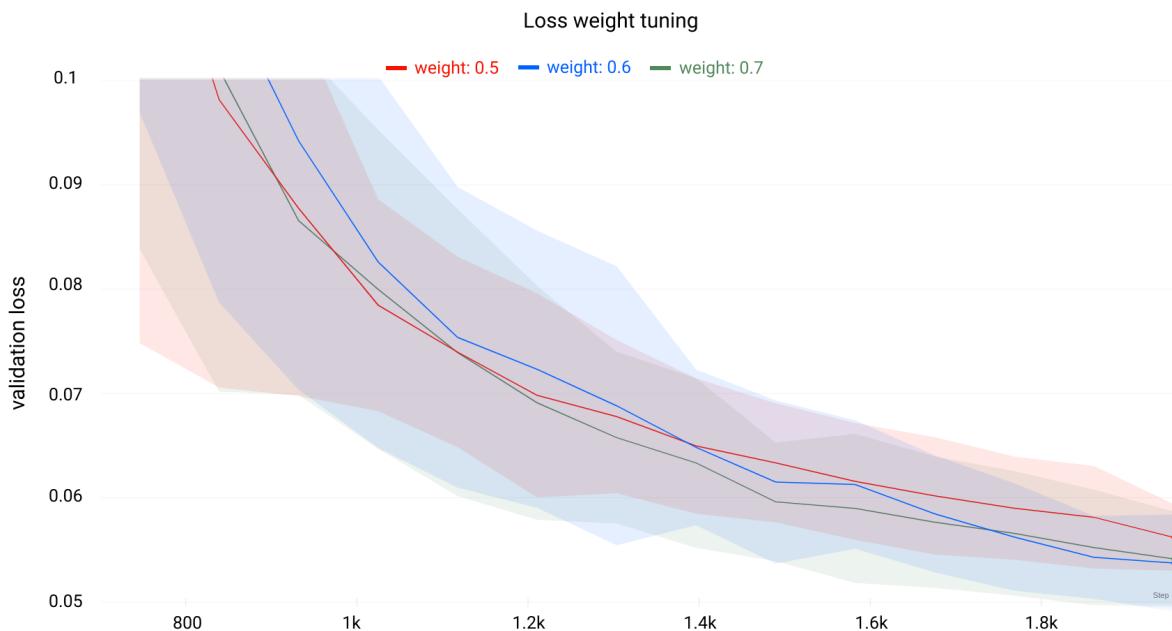
Pro  $w$  jsem zvolil hodnoty: 0.5, 0.6, 0.7. Sítě jsem seskupil podle hodnoty  $w$ . Neuronové sítě dosahovaly nejnižší validační chyby při  $w = 0.6$  (viz obrázek 16).

Poté jsem hledal vliv enkovérů. Vytrénoval jsem 132 neuronových sítí. Po vybrání jen těch

úspěšných (validační chyba menší než 0.1), zbylo 61 neuronových sítí. Hodnotu validační chyby jsem zprůměroval podle enkodéru. Nejúspěšnejší se ukázal být ResNet-34 enkodér (viz obrázek 17). Jelikož je kvalita učení silně závislá na parametrech *batch size* a *learning rate*, tak jsem při hledání správného enkodéru optimalizoval i tyto parametry. ResNet-34 enkodér dosahoval nejlepších výsledků v případě *batch size* = 16 a *learning rate* = 0.001 (viz obrázek 18).



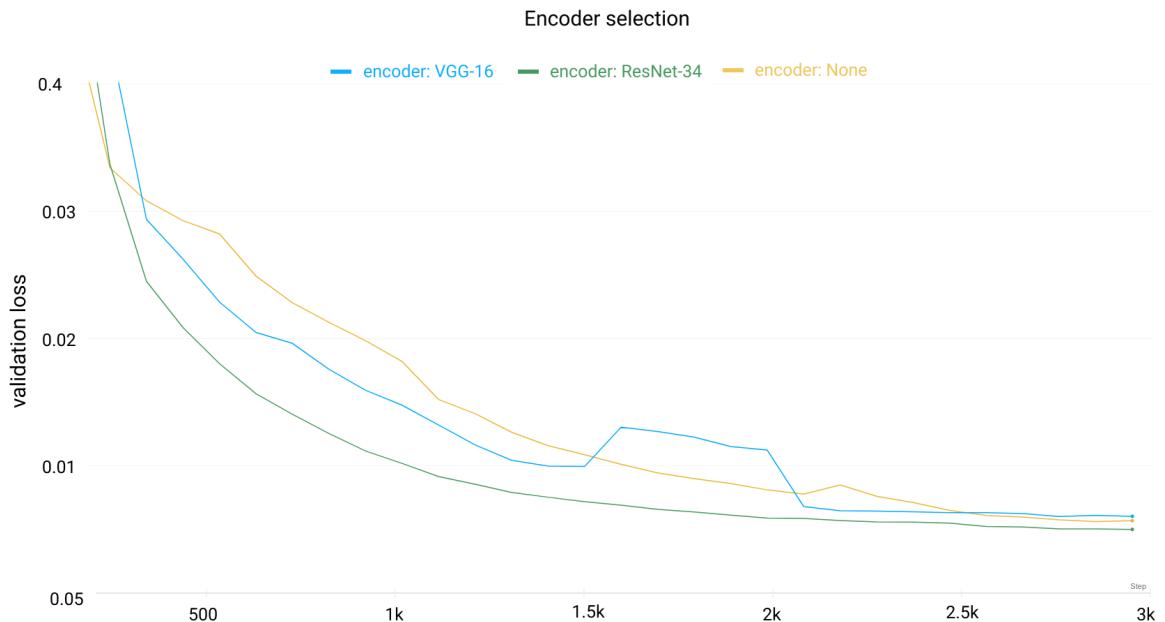
Obrázek 15: Průměrná validační úspěšnost  $\text{IoU}_m$  neuronových sítí s různými hodnotami  $\alpha$ . Světlou barvou je vynáčena směrodatná odchylka pro hodnoty  $\alpha$ .



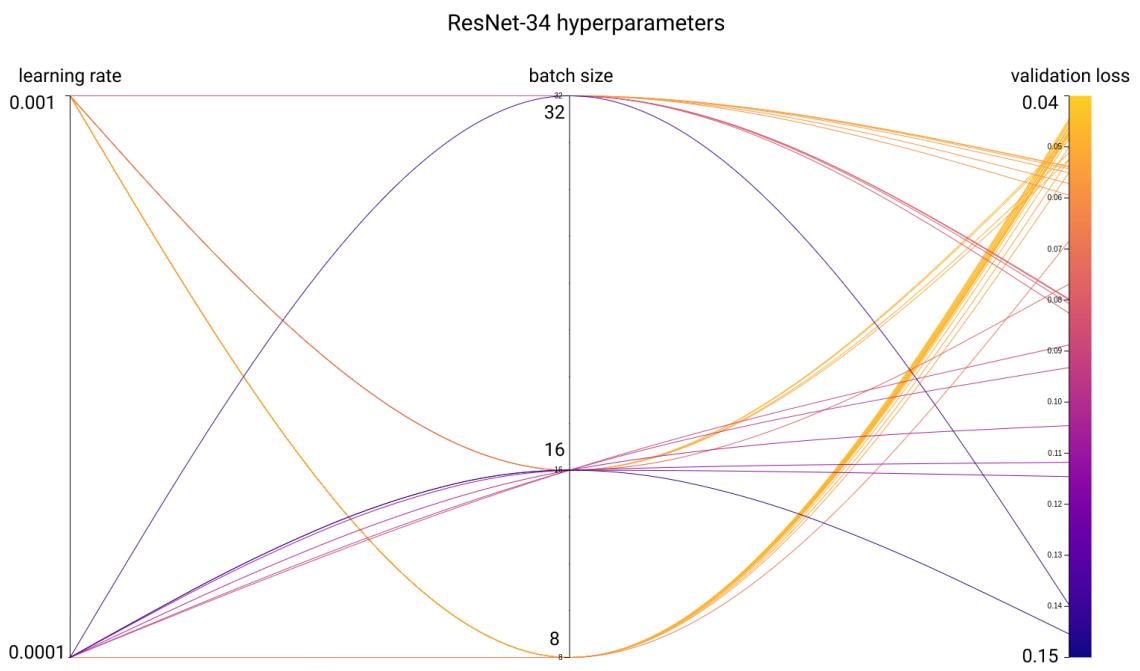
Obrázek 16: Průměrná validační chyba neuronových sítí s různými hodnotami  $w$ . Světlou barvou je vynáčena směrodatná odchylka pro hodnoty  $w$ .

Po nalezení správných hyperparametrů jsem vytrénoval síť na neomezeném počtu epoch (dokud

se zlepšovala validační úspěšnost). Tato síť zlepšila segmentaci oproti původní metodě téměř o 16 procentních bodů (viz tabulka 19). Ukázka masek vytvořených metodou popsané v této práci je na obrázku 20.



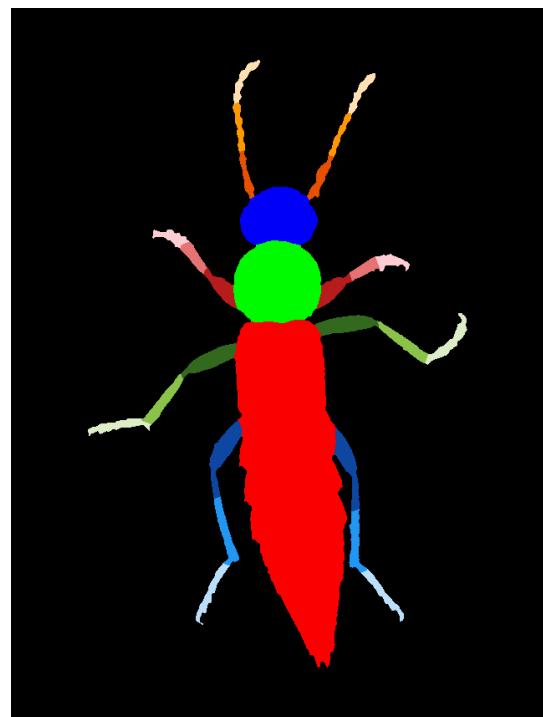
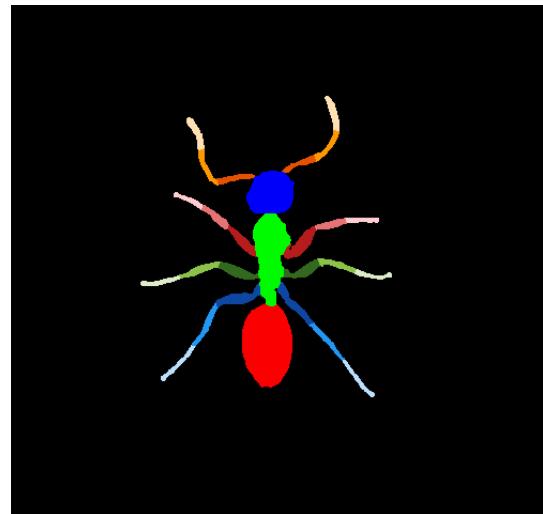
Obrázek 17: Průměrná validační chyba s různými enkodéry.



Obrázek 18: Vliv *batch size* a *learning rate* na validační chybu ResNet-34 enkodéru.

	<b>Mean Pixel-wise</b>	<b>Mean IoU</b>	<b>Binary IoU</b>	<b>Průměr</b>
Metoda v této práci	99.82	90.31	93.29	94.47
Segmentační metoda v předchozí práci	98.99	52.42	85.84	79.08

Table 19: Porovnání úspěšnost metod na testovací množině. Hodnoty jsou vyjádřeny v procentech.



(a) Vstupní obrázky.

(b) Masky vytvořené metodou v této práci.

Obrázek 20: Ukázka cílových masek vytvořených metodou v této práci.

## 6 Diskuze

Pro segmentaci obrazu existuje spousta metod. Já jsem se v této práci věnoval konvolučním neuronovým sítí. Jedním z hlavních faktorů při použití neuronových sítí je zvolení správné architektury. Rozhodl jsem se pro U-Net architekturu, která je často používaná pro segmentaci obrazu a je na ní provedena spousta výzkumů o různých vylepšeních [24], [25]. Pro vytrénování úspěšné neuronové sítě je také potřeba nalézt správnou kombinaci hyperparametrů a správně tuto síť natrénovat. Protože trénování neuronových sítí je výpočetně velmi náročné, rozhodl jsem se tyto hyperparametry optimalizovat po skupinách a jako správnou kombinaci jsem použil průměr výsledků pro různé hodnoty hyperparametrů. Snížil se tím počet neuronových sítí potřebných k vytrénování (320 – přibližně 200 hodin trénování), ale je možné, že jiná kombinace parametrů mohla dosahovat lepších výsledků. Neuronová síť byla vytrénována na datech, kde členovec byl natočen hlavou směrem nahoru. Byla by nefunkční, pokud by byl členovec natočen hlavou dolů. Vytrénovat neuronovou síť, která by byla schopna správně určit i obrácené obrázky by bylo velmi náročné, protože někteří členovci jsou téměř symetričtí kolm horizontální osy. Ostatní metody v této práci také spolehají na to, že členovec byl natočen hlavou nahoru a vyfocen z vrchu. Mnou vyvinutá neuronová síť dokázala na každém obrázku z testovací množiny nalézt hledaný objekt. V některých případech zařadila více instancí jednoho objektu než mělo být. Dělo se tak kvůli nečistotám na pozadí. Proto jsem k potlačení těchto nečistot zkusil použít metodu Non-Local Means Denoising [26]. Ukázalo se ale, že měla negativní vliv na úspěšnost neuronové sítě. Nejspíše proto, že vyhlazovala také přechod mezi pozadím a objektem, což mělo negativní vliv na konvoluce. Dalším problémem byla situace, kdy členovec splýval s pozadím nebo byl průsvitný. Na poskytnutých obrázcích se tato situace vyskytovala jen ojediněle, tím pádem neuronová síť neměla dost dat, aby se tuto situaci naučila.

Úspěšnost v určení správného tvaru těla, délky těla a spárování končetin závisela na úspěšném nalezení osy symetrie. Úspěšnost nalezení osy se snižovala, pokud byl objekt moc krátký nebo nebyl vyfocen přímo z vrchu. Zkusil jsem použít i metodu momentů [27], ale ta měla horší výsledky na mé datové množině. Jako řešení bych navrhl při určení osy symetrie objektu brát v potaz i ostatní objekty a nepovažovat těžiště objektu za bod osy.

Dělení končetin na tři stejně dlouhé části bylo závislé na správném určení bodů s geodetickou vzdáleností nula. To se mohlo stát nepřesné v případě, že končetina byla zanořená do těla a sdílela s tělem více bodů, než jen ty počáteční. Možným řešením by bylo nedávat všem sousedícím bodům s tělem geodetickou vzdálenost rovnou nule.

Klíčovým bodem při počítání histogramu barevných kanálů je správné určení masky odlesků. Nesprávnému určení masky odlesků docházelo v případě velkého barevného rozdílu (např. černá a bílá) na jednom objektu, kdy byly všechny světlé části považovány za odlesk. Problémem odlesků se zabývá spousta článků [28], [29], ale zatím neexistuje univerzální řešení a metody navržené v těchto článcích neposkytly dostatečně dobrý výsledek na mé datové množině.

## **7 Závěr**

Pomocí neuronových sítí se mi povedlo splnit hlavní cíl této práce tj. zlepšit metodu segmentace studovaného objektu. Zlepšení je jak v samotném nalezení studovaného objektu na obrazu, tak i v rozdelení na menší regiony. Má metoda segmentace není přímo závislá na tvaru těla na rozdíl od předchozí metody segmentace, čímž je použitelná i na jiné členovce.

Dále jsem vyvinul metody na charakteristiku členovců z obrázku, které zahrnují určení tvaru těla, spočítání rozměrů a barevnou charakteristiku.

## 8 Použitá literatura

- [1] Moya-Laraño J. et al. (2013). „Spider Research in the 21st Century: Trends & Perspectives“. In: ed. D. Penney. Siri Scientific Press. Kap. Evolutionary Ecology. Linking traits, selective pressures and ecological functions, s. 112–153.
- [2] Pekár J. (2017). „Sledování hmyzu ve videu“. SOČ. Gymnázium Brno, Vídeňská, příspěvková organizace, Vídeňská 55/47, 639 00 Brno.
- [3] Ježek J. (2015). „Měření podobnosti pavouků a mravenců“. Dipl. Masarykova univerzita, Fakulta informatiky.
- [4] O’Mahony N. et al. (2019). „Deep learning vs. traditional computer vision“. In: *Science and Information Conference*, s. 128–144.
- [5] Perez L. a J. Wang (2017). „The effectiveness of data augmentation in image classification using deep learning“. arXiv preprint arXiv:1712.04621.
- [6] Ronneberger O., P. Fischer a T. Brox (2015). „U-net: Convolutional networks for biomedical image segmentation“. In: *International Conference on Medical image computing and computer-assisted intervention*. Cham: Springer, s. 234–241.
- [7] Agarap A. F. (2018). „Deep learning using rectified linear units (relu)“. arXiv preprint arXiv:1803.08375.
- [8] Srivastava N. et al. (2014). „Dropout: a simple way to prevent neural networks from overfitting“. In: *The journal of machine learning research* 15.1, s. 1929–1958.
- [9] Ulyanov D., A. Vedaldi a V. Lempitsky (2016). „Instance normalization: The missing ingredient for fast stylization“. arXiv preprint arXiv:1607.08022.
- [10] Simonyan K. a A. Zisserman (2014). „Very deep convolutional networks for large-scale image recognition“. arXiv preprint arXiv:1409.1556.
- [11] He K. et al. (2016). „Deep residual learning for image recognition“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, s. 770–778.
- [12] Russakovsky Olga et al. (2015). „ImageNet Large Scale Visual Recognition Challenge“. In: *International Journal of Computer Vision (IJCV)* 115.3, s. 211–252.
- [13] Robbins H. a S. Monro (1951). „A stochastic approximation method“. In: *The annals of mathematical statistics* 22.3, s. 400–407.
- [14] Sørensen T. (1948). „A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons“. In: *Kongelige Danske Videnskabernes Selskab* 5.4, s. 1–34.
- [15] Tversky A. (1977). „Features of similarity“. In: *Psychological review* 84.4, s. 327–352.
- [16] Jaccard P. (1901). „Étude comparative de la distribution florale dans une portion des Alpes et des Jura“. In: *Bull Soc Vaudoise Sci Nat* 37, s. 547–579.
- [17] Dijkstra E. W. (1959). „A note on two problems in connexion with graphs“. In: *Numerische mathematik* 1.1, s. 269–271.
- [18] Zhang T. Y. a C. Y. Suen (1984). „A fast parallel algorithm for thinning digital patterns“. In: *Communications of the ACM* 27.3, s. 236–239.
- [19] Bradski G. (2000). „The OpenCV Library“. In: *Dr. Dobb’s Journal of Software Tools*.

- [20] Paszke Adam et al. (2019). „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems 32*. Ed. H. Wallach et al. Curran Associates, Inc., s. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [21] Biewald L. (2020). *Experiment Tracking with Weights and Biases*. Software available from wandb.com. URL: <https://www.wandb.com/>.
- [22] *Qt* (2020). URL: [www.qt.io](http://www.qt.io).
- [23] Salehi S. S. M., D. Erdogan a A. Gholipour (2017). „Tversky loss function for image segmentation using 3D fully convolutional deep networks“. In: *International workshop on machine learning in medical imaging*, s. 379–387.
- [24] Li X. et al. (2018). „H-DenseUNet: hybrid densely connected UNet for liver and tumor segmentation from CT volumes“. In: *IEEE transactions on medical imaging* 37.12, s. 2663–2674.
- [25] Weng Y. et al. (2019). „NAS-Unet: Neural architecture search for medical image segmentation“. In: *IEEE Access* 7, s. 44247–44257.
- [26] Buades A., B. Coll a J. M. Morel (2011). „Non-local means denoising“. In: *Image Processing On Line* 1, s. 208–212.
- [27] Burger W. a M. J. Burge (2008). *Digital image processing: an algorithmic introduction using Java*. New York, USA: Springer.
- [28] Yang Q., J. Tang a N. Ahuja (2014). „Efficient and robust specular highlight removal“. In: *IEEE transactions on pattern analysis and machine intelligence* 37.6, s. 1304–1311.
- [29] Fu G. et al. (2019). „Specular Highlight Removal for Real-world Images“. In: *In Computer Graphics Forum* 38.7, s. 1304–1311.

## 9 Seznam obrázků a tabulek

1	Pavouk (vlevo) napodobující mravence (vpravo). Převzato z <a href="https://upload.wikimedia.org/wikipedia/commons/e/ea/Ant_and_jumping_spider_Gorongosa_National_Park%2C_Mozambique.jpg">https://upload.wikimedia.org/wikipedia/commons/e/ea/Ant_and_jumping_spider_Gorongosa_National_Park%2C_Mozambique.jpg</a> . . . . .	7
2	Ukázka masek vytvořených předchozí prací. . . . .	9
3	Ukázka poskytnutých dat. . . . .	11
4	Princip fungování konvoluční vrstvy. Převzato z <a href="https://commons.wikimedia.org/wiki/File:Konvoluce_2rozm_diskretni.jpg">https://commons.wikimedia.org/wiki/File:Konvoluce_2rozm_diskretni.jpg</a> . . . . .	13
5	Vlevo vstupní obraz, vpravo po alpikaci $2 \times 2$ max-pooling. Převzato z <a href="https://embarc.org/embarc_mli/doc/build/html/MLI_kernels/pooling_max.html">https://embarc.org/embarc_mli/doc/build/html/MLI_kernels/pooling_max.html</a> . . . . .	14
6	Architektura konvoluční neuronové sítě U-Net. Zeleně označena smršťovací část, modře oblast zúžení a růžově expanzní oblast. Převzato z <a href="https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/u-net-architecture.png">https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/u-net-architecture.png</a> . . . . .	15
7	Ukázka rozdělení končetin na menší regiony. . . . .	20
8	Červeně jsou vyznačené spoje, ve kterých se může mravenec ohýbat. . . . .	21
9	Grafické uživatelské prostředí pro úpravu osy symetrie. . . . .	22
10	Ukázka osy symetrie a tvaru těla. . . . .	23
11	Metody měření délky hlavové části. . . . .	23
12	Červená čára vyznačuje kostru končetiny. . . . .	24
13	Ukázka masky odlesků. . . . .	24
14	Průměrná validační chyba neuronových sítí s různými hodnotami $\alpha$ . Světlou barvou je vynáčena směrodatná odchylka pro hodnoty $\alpha$ . . . . .	26
15	Průměrná validační úspěšnost $IoU_m$ neuronových sítí s různými hodnotami $\alpha$ . Světlou barvou je vynáčena směrodatná odchylka pro hodnoty $\alpha$ . . . . .	27
16	Průměrná validační chyba neuronových sítí s různými hodnotami $w$ . Světlou barvou je vynáčena směrodatná odchylka pro hodnoty $w$ . . . . .	27
17	Průměrná validační chyba s různými enkodéry. . . . .	28

18	Vliv <i>batch size</i> a <i>learning rate</i> na validační chybu ResNet-34 enkodéru. . . . .	28
19	Porovnání úspěšnost metod na testovací množině. Hodnoty jsou vyjádřeny v procentech. . . . .	29
20	Ukázka cílových masek vytvořených metodou v této práci. . . . .	29