Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

# DIPLOMOVÁ PRÁCE

Metody strojového učení pro zpracování ekologických dat

Rozpoznávání a klasifikace fotografií čolka velkého

2019

Bc. Radomír Nádvorník

Vedoucí práce: RNDr. Eduard Bartl, Ph.D.

Studijní obor: Informatika, prezenční forma

**Bibliografické údaje**

| | |
|---|---|
| Autor: | Bc. Radomír Nádvorník |
| Název práce: | Metody strojového učení pro zpracování ekologických dat (Rozpoznávání a klasifikace fotografií čolka velkého) |
| Typ práce: | diplomová práce |
| Pracoviště: | Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci |
| Rok obhajoby: | 2019 |
| Studijní obor: | Informatika, prezenční forma |
| Vedoucí práce: | RNDr. Eduard Bartl, Ph.D. |
| Počet stran: | 50 |
| Přílohy: | 1 CD |
| Jazyk práce: | český |

**Bibliograhic info**

| | |
|---|---|
| Author: | Bc. Radomír Nádvorník |
| Title: | Machine learning in ecological data processing (Recognition and classification of Triturus cristatus images) |
| Thesis type: | master thesis |
| Department: | Department of Computer Science, Faculty of Science, Palacký University Olomouc |
| Year of defense: | 2019 |
| Study field: | Computer Science, full-time form |
| Supervisor: | RNDr. Eduard Bartl, Ph.D. |
| Page count: | 50 |
| Supplements: | 1 CD |
| Thesis language: | Czech |

## Anotace

*Práce pojednává o webové aplikaci, která byla vytvořena pro účely rozpoznávání a klasifikace fotografií čolka velkého. Aplikace byla vytvořena na námět a ve spolupráci s katedrou ekologie a životního prostředí na přírodovědecké fakultě v Olomouci. První část textu se zaměřuje na popis sběru dat a způsob jakým jsou data vyhodnocována. Další část textu navazuje popisem metod strojového učení a metod zpracování obrazu, které byly využity při implementaci a poslední část práce je věnována podrobnému technickému popisu řešení.*

## Synopsis

*The thesis discuss a web application designed for recognition and classification of Triturus cristatus images. The application was created on the suggestion and in cooperation with the Department of Ecology and the Environment at the Faculty of Science in Olomouc. The first part of the text focuses on the description of the data collection and the way the data are evaluated. The second part of the text describes the methods of machine learning and image processing used during the implementation and the last part of the thesis is devoted to the detailed technical description of the application.*



**Klíčová slova:** Strojové učení; čolek velký; webová aplikace; Google Cloud

**Keywords:** Machine learning; Great Crested Newt; application; Google Cloud

Rád bych poděkoval svému vedoucímu diplomové práce RNDr. Eduardu Bartlovi Ph.D., RNDr. Tomáši Fürstovi Ph.D. a Mgr. Lukáši Weberovi za užitečné rady, věcné připomínky a vstřícnost při konzultacích.

*Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.*

datum odevzdání práce podpis autora

# Obsah

# Seznam obrázků



# Seznam tabulek



# Seznam zdrojových kódů

# 1 Úvod

Cílem práce je navrhnout, popsat a implementovat vhodný algoritmus na rozpoznávání charakteristických rysů čolka velkého z fotografií pořízených při jeho odchytu. Tyto charakteristiky se využívají při hledání stejných jedinců na různých fotografiích. Pomocí metody zpětného odchytu je možné určit odhad velikosti populace konkrétního biologického druhu v dané lokalitě. Potřebným vstupem této metody je právě počet shodných odchycených jedinců, který je získán porovnáváním fotografií. Hledání stejných jedinců na různých fotografiích se aktuálně provádí manuálně. Porovnávají se vždy nově pořízené fotografie se všemi předchozími a jedná se tak o časově velmi náročnou činnost. Nalezení vhodného algoritmu na rozpoznávání charakteristických rysů čolka velkého a také způsobu automatického srovnávání fotografií, by mohlo vést ke značné časové úspoře při určování odhadu velikosti populace.

Navržené řešení bude implementováno ve webové aplikaci, která bude poskytovat strukturované zobrazení nahraných fotografií, jejich kompletní analýzu a porovnávání a také výpočet odhadu velikosti populace pomocí metody zpětného odchytu. Aplikace bude zohledňovat doposud získané zkušenosti během výzkumu čolka velkého. S využitím metod strojového učení bude možné určit kvalitu nahrávaných fotografií a také rozpoznat pohlaví jedinců čolka z fotografií. Vybrané části aplikace budou navrženy tak, aby mohly být využity při případném rozšíření pro další biologické druhy, u kterých se využívají podobné principy porovnávání charakteristických rysů.

Text práce seznamuje čtenáře s využívanými metodami při odchytu čolka velkého, způsobu značení jedinců a výpočtu odhadu velikosti populace. Dále text popisuje metody strojového učení a metody zpracování obrazu, které jsou využity při hledání a rozpoznávání charakteristických rysů čolka a při srovnávání s ostatními fotografiemi. Následuje uživatelská příručka, popis využitých technologií a programátorská dokumentace. Nedílnou součástí textu je také popis platformy Google Cloud, která byla využita při vývoji aplikace a nyní zajišťuje její hostování, včetně centrálního uložení dat.

# 2 Odchyt čolka velkého

Katedra ekologie a životního prostředí na Přírodovědecké fakultě Univerzity Palackého v Olomouci se mimo jiné zabývá i výzkumem čolka velkého. Oblastí zájmu je jeho způsob života, charakteristické rysy, výskyt, migrace a s tímto související odhad velikosti populace. K určení odhadu velikosti populace je využívána metoda zpětného odchytu, které se budeme v této kapitole věnovat. Uvedeme také způsoby sběru dat a jejich zpracování pro potřeby určení odhadu populace. V návaznosti na tato témata uvidíme i případné výhody aplikace pro rozpoznávání charakteristických rysů čolka velkého, která byla připravována ve spolupráci se zmíněnou katedrou ekologie a životního prostředí.

## 2.1 Čolek velký

Čolek velký - *Triturus cristatus* (*Great Crested newt*) je obojživelník patřící do řádu mloků. Jedná se o druh tzv. naturový, silně ohrožený a chráněný dle evropského práva. Tělo dospělých jedinců dle [27] dorůstá 150 až 200 mm a délka ocasu je přibližně stejná jako délka těla s hlavou. Čolek velký je rozšířený ve větší části Evropy.

Důležitým znakem jsou tmavé skvrny na žlutém až oranžovém břiše. Tyto skvrny jsou využívány při porovnávání jedinců čolka velkého. Samice je možné od samců rozeznat díky načervenalé a zploštělé kloace a ocasu, který má samice s načervenalým nebo oranžovým pruhem. Samec má kloaku spíše tmavou, stejně jako ocas, který nemá světlý pruh. V období rozmnožování žijí čolci ve vodě a samec je v této době význačný svým zubatým kožním hřebenem v pánevní oblasti. Mladí jedinci čolka velkého jsou po vývinu z larvy a přesunu na souš téměř stejně velcí jako dospělí jedinci, ale chybí jim výrazné skvrny na břiše a ostatní odlišující znaky samce a samice. Z tohoto důvodu není možné až do dospělosti určit pohlaví jedince. Podrobnější informace lze nalézt například v knize [17]. Pro námi řešený problém je popis uvedený výše dostatečný.

## 2.2 Metoda zpětného odchytu

Pro odhad velikosti populace nebo podskupin této populace se používá metoda zpětného odchytu. Ačkoli má tato metoda určitá omezení, je dle [20] užitečná pro odhad počtu ohrožených a nepolapitelných jedinců. Metoda zpětného odchytu je v biologických a ekologických oborech velmi využívaná.

Základní myšlenka spočívá v odchycení několika jedinců zkoumaného zvířete, jejich označením, vypuštěním zpět do volné přírody a znovu odchycením. Při dalším odchytu[1] jsou tady spočítáni zvlášť jedinci, kteří byli během prvního odchytu označeni a zvlášť jedinci, kteří byli odchycení poprvé. Metodu je možné rozšířit na více, než dva po sobě jdoucí odchyty a tím je odhad populace, který tato metoda

[1] Odchytem je myšlen časový úsek, během kterého probíhá odchycení a označení odchycených jedinců na určitém místě.

Obrázek 1: Fotografie samice (vlevo) a samec čolka velkého (vpravo).

poskytuje, přesnější. Důležitá je volba způsobu označení chycených jedinců, aby bylo označení trvanlivé a navíc umožňovalo rozlišit mezi jednotlivými odchyty. K tomuto může posloužit například pořizování fotografií chycených jedinců, jak uvidíme v kapitole 2.3.

### 2.2.1 Výpočet odhadu populace

Vezměme již popsané údaje a označme je následovně:

- $N$... hledaná velikost populace,
- $M_1$... počet chycených, označených a znovu vypuštěných jedinců v prvním odchytu,
- $M_2$... počet chycených jedinců při druhém odchytu,
- $R$... počet znovu chycených (tzv. *recaptured*) jedinců, tedy těch, kteří byli chyceni při prvním i druhém odchytu.

Výpočet odhadu populace je založen na porovnání poměrů, kde poměr znovu chycených jedinců ($R$) ze všech chycených při druhém odchytu ($M_2$) je roven poměru jedinců chycených na začátu ($M_1$) vůči počtu celé populace ($N$). Jednoduchou úpravou vztahu dostáváme vztah (2) pro výpočet odhadu velikosti populace:

$$\frac{R}{M_2} = \frac{M_1}{N_{Lin}}, \tag{1}$$

$$N_{Lin} = \frac{M_1 \cdot M_2}{R} \tag{2}$$

Uvedený vztah se nazývá Lincolnův index (*Lincoln index*). Metoda je také často označována jako *Lincolnon-Petersen method/estimator*, podle Johannese Petersena, který ji poprvé využil ve spojitosti s metodou zpětného odchytu. Ze vztahu lze jednoduše vidět, že čím větší je číslo $R$, tím menší je velikost celé populace. Tato skutečnost je plně v souladu s naším předpokladem. Teorie byla převzata z knihy [24], kde jsou k nalezení i další podrobnosti. Metody výpočtu odhadu velikosti populace předpokládají, že:

- populace se nemění mezi jednotlivými odchyty (populace je uzavřená),
- všichni jedinci mají stejnou pravděpodobnost, že budou odchyceni,
- skutečnost, že byl odchycen jedinec během prvního odchytu nemění pravděpodobnost možnosti odchytu při druhém z odchytů,
- samotné označení jedinců během odchytů nezmizí,
- označení jedinci budou při druhém odchycení správně započítáni do skupiny *recaptured* ($R$).

Podle [12] je Lincolnův index přesný pro velké (nekonečné) množiny jedinců. Na menších vzorcích je poněkud nepřesný. Pro tyto případy můžeme získat lepší výsledek díky tzv. Chapmanovu odhadu (*Chapman estimator*), který je definován následovně:

$$N_{Cha} = \frac{(M_1 + 1) \cdot (M_2 + 1)}{R + 1} - 1 \tag{3}$$

Vztahy uvedené výše poskytují odhad velikosti populace na základě dvou odchytů. Pro ještě přesnější odhad je možné využít tzv. *Schnabel index* (Schnabelova metoda), který zohledňuje libovolný počet po sobě jdoucích odchytů ([18]). Rozšiřme značení uvedené na začátku této kapitoly následovně:

- $m$... celkový počet odchytů,
- $M_i$... celkový počet chycených jedinců během odchytu $i$,
- $T_i$... celkový počet označených jedinců při předchozích odchytech,
- $R_i$... počet znovu chycených (tzv. *recaptured*) jedinců během odchytu $i$, tedy těch, kteří byli chyceni také při některém předchozím odchytu.

*Schnabel index* je přímým rozšířením Lincolnova indexu. Matematicky jej můžeme zapsat tímto způsobem:

$$N_{Sch} = \frac{\sum_{i=1}^{m} M_i \cdot T_i}{\sum_{i=1}^{m} R_i} \tag{4}$$

| **i** | $\mathbf{M_i}$ | $\mathbf{R_i}$ | $\mathbf{T_i}$ | $\mathbf{N_{Lin}}$ | $\mathbf{N_{Cha}}$ | $\mathbf{N_{Sch}}$ |
|---|---|---|---|---|---|---|
| 1 | 100 | 0 | 0 | - | - | - |
| 2 | 85 | 15 | 100 | 566 | 535 | 566 |
| 3 | 105 | 25 | 170 | 714 | 696 | 658 |

Tabulka 1: Tabulka odhadů populace pro jednotlivé metody.

Pro ilustraci uveďme konkrétní příklad výpočtu. Odhady populace na základě popsaných metod jsou uvedeny v tabulce 1. V příkladu byly provedeny tři odchyty. Při prvním bylo chyceno a označeno 100 jedinců. Při druhém bylo chyceno 85 jedinců a z toho 15 bylo označeno při prvním odchytu. Hodnota $T_2$ se rovná počtu chycených jedinců v prvním odchytu. Vidíme, že Lincolnova metoda a metoda Schnabelové poskytují v této části stejný odhad velikosti populace. Poznamenejme, že pokud je výsledkem výpočtu desetinné číslo, za výsledek považujeme jeho dolní celou část.

Při třetím odchytu bylo odchyceno 105 jedinců, z toho 25 označených z celkových 170 označených jedinců v předchozích odchytech. *Schnabel index* zohledňuje při výpočtu odhadu druhou iteraci odchytu, tedy to, že bylo z 85 odchycených 15 označených. Vztah (2) ani (3) toto neumožňují a proto je jich výsledek obecně méně přesný při uvažování více odchytů.

Doposud popsané metody zpětného odchytu poskytují odhad populace pro tzv. uzavřené populace a pracují pouze s počtem chycených a označených jedinců. Pro značení může být použit stejný typ značky. Jolly-Seberova metoda (*Jolly-Seber method*) zohledňuje při výpočtu i fakt, kdy byl daný jedinec odchycen naposledy a s její využitím lze odhadhnout velikost tzv. otevřené populace. Důležité je využít specifické označení pro každý odchyt.

$$N_{Jol} = \frac{P_i}{\alpha_i}, \tag{5}$$

$$P_i = \frac{(s_i + 1) \cdot Z_i}{S_i + 1} + R_i, \qquad \alpha_i = \frac{R_i + 1}{M_i + 1} \tag{6}$$

- $P_i$... odhad velikosti označené populace,
- $\alpha_i$... odhad poměrné části označených jedinců,
- $s_i$... počet všech vypuštěných jedinců po odchytu $i$ ($M_i$ např. bez úmrtí),
- $Z_i$... počet chycených jedinců před odchytem $i$, nechycených během odchytu $i$, ale chycených v některém pozdějším,
- $S_i$... počet $s_i$ chycených v některém pozdějším odchytu,
- $R_i$, $M_i$... definované stejně jako u předchozích metod.

Jolly-Seberova metoda navíc zohledňuje tzv. vstup do populace a přežívací funkci. Mimo stanovení odhadu velikosti populace lze z uvedeného vztahu (5) vypočítat také odhad dalších vlastností populace. Například pravděpodobnost přežití, přírustek populace nebo míru populačních změn. Převzato z [16].

## 2.3 Značení jedinců

V kapitole 2.2 jsme se věnovali teoretickému a matematickému popisu metody zpětného odchytu, která se využívá pro výpočet odhadu velikosti populace. Vstupem popsaných metod je počet jedinců zvířete. Musíme je být tedy schopni odlišit. Častým způsobem je fyzický odchyt určité skupiny jedinců, jejich spočítání a následné vypuštění. Způsob se však může lišit v závislosti na druhu zvířete. Jedinci čolka velkého jsou pro tyto účely chytáni do tzv. živolovných pastí nebo loveni tzv. prolovením tůně jejich výskytu.

Po odchycení je potřeba každého jedince vhodným způsobem označit. Podobně jako odchyt se i označení líší v závisloti na druhu zvířete. Mnohdy je možné na zvíře umístit štítek nebo kroužek s jeho označením. Někdy se označení umísťuje barvou přímo na část těla zvířete. U čolků nebyla žádná z těchto možností vyhovující. V minulosti se chycení jedinci odlišovali stříháním článků prstů. Mimo možný negativní vliv na jedince bylo toto řešení neoptimální i v souvislosti s využitím v metodě zpětného odchytu vyžadující časově specifické značení (například Jolly - Seberova metoda [16]).

Pokud některá z charakteristických vlastností nebo znaků zvířete umožňuje jeho jednoznačnou identifikaci v celé populaci, nemusí být jedinci nijak dále označováni. V těchto případech je potřeba pouze zaznamenat jednoznačně rozlišující charakteristiky a ty následně vzájemně porovnávat. Pro čolka velkého splňují tuto vlastnost výrazné tmavé skvrny na jeho břiše. Pro možnost porovnávání se při odchytu pořizují fotografie.

Každý chycený jedinec je umístěn na žluté nebo zelené pozadí otočený břichem se skvrnami nahoru. Pro lepší manipulaci při focení se čolek položí na průhlednou podložku a přidrží se navlhčenou houbičkou. Příklad fotografií je na obzázku 1. Fotografie jsou pořizovány přímo v terénu při odchytu. Z toho vyplývá, že některé fotografie mohou být méně kvalitní a obtížněji porovnatelné s ostatními. V následující kapitole je popsán způsob zpracování a vyhodnocení pořízených fotografií.

## 2.4 Manuální porovnávání fotografií

Klíčovou částí v procesu stanovení odhadu populace při použití metody zpětného odchytu je správné určení počtu již chycených jedinců. U technik, kdy se používá označení štítkem, kroužkem nebo jinou značkou, která je unikátní, je snadné určit o kolikáté odchycené daného jedince se jedná. Proces určení počtu znovu chycených čolků je složitější právě kvůli tomu, že jsou foceni. Při chycení čolka tedy ihned nevíme, zda byl již někdy chycen. Zjistíme to porovnáním každé nové

Obrázek 2: Porovnávání skvrn člověkem.

fotografie se všemi předchozími. Zvlášť se mezi sebou porovnávají samci a zvlášť samice. Určení pohlaví předchází právě této fázi porovnávání. Lehce vidíme, že čím více je odchytů provedeno a čím více jedinců je při každém z odchytu vyfoceno, tím obtížnější se porovnávání stává a zároveň se zvyšuje pravděpodobnost chycení již vyfoceného čolka.

Srovnávání jednotlivých fotografií aktuálně provádí člověk a zjednodušení tohoto procesu je předmětem této práce. V první fázi jsou čolci roztřízeni na dvě skupiny podle pohlaví. Vyřazena z procesu jsou mláďata, u kterých nelze pohlaví ještě určit. Samice se od samců liší ve zbarvení kloaky. U samců je kloaka v období rozmnožování černá a zduřelá, u samic je více zploštělá a načervenalá. Samec má navíc na ocase modro stříbrný pruh, u samic je pruh oranžový nebo načervenalý, jak již bylo zmíněno v kapitole 2.1.

Dále je nutné porovnat čolky v rámci těchto skupin. Unikátním identifikátorem je tvar, poloha a velikost skvrn na břiše. V praxi se ukázalo nejsnažší nalézt nevšední skvrnu nebo skupinu skvrn a tuto následně zkusit najít na ostatních fotografiích. Příklad je uveden na obrázku 2. Modrým kruhem jsou zde označeny specifické skvrny, které se s vysokou pravděpodobností u žádného jiného jedince vyskytovat nebudou. Po nalezení některých společných charakteristických skvrn, dokážeme již jednoduše ověřit, že se i zbývající skvrny shodují. Na příkladu vidíme fotografie stejného jedince, které byly pořízeny při dvou různých odchytech, konkrétně 13.5.2015 a 29.5.2015. Můžeme si také díky oranžovému pruhu na ocase a načervernalé kloace všimnout, že se jedná o samici čolka velkého. Pokud není

nalezena žádná shoda, jedná se o nového, doposud nechyceného jedince. Přesto, že je na fotografiích stejný jedinec čolka, mají tyto fotografie mnoho odlišností. Na první pohled můžeme na příkladu 2 vidět, že nejsou zajištěny stejné světelné podmínky, pozadí ani pozice porovnávaného čolka. Tím dochází k deformaci, zakrytí nebo naopak spojení některých skvrn. Tyto problémy budou řešeny dále v textu ve spojitosti s popisem metod využitých při implementaci aplikace.

# 3 Umělá inteligence

Umělá inteligence (*Artificial Intelligence*, zkr. AI) je oblast informatiky, která se zabývá otázkou „inteligence“ počítačů a strojů obecně. Inteligenci lze definovat mnoha způsoby. Pro naše účely je dostatečné intuitivní chápání tohoto pojmu. Lidská inteligence je v kontextu tohoto oboru dle [13] označována často jako přirozená inteligence a snaha o simulaci lidského myšlení počítačem je právě předmětem inteligence umělé. Význam slova inteligence se v průběhu let mění a věci, které jsou „inteligentní“ se časem stávají běžné, lépe pochopitelné a s tím je spojen i vývoj celé této oblasti informatiky.

Podobně jako jsou rozsáhlé obory zabývající se studiem lidského vnímání, myšlení, logikou a dalšími oblastmi spojovanými s inteligencí, je i umělá inteligence velmi obecný a rozsáhlý pojem. Umělá inteligence zahrnuje například řešení problémů, automatické plánování, reprezentaci znalostí, strojové učení, pohyb, vnímání a mnoho dalšího (převzato z [13]). V následující kapitole se budeme zajímat o metody strojového učení a umělé neuronové sítě, které jsou využívány při klasifikaci čolků.

## 3.1 Metody strojového učení

Strojového učení (*Machine Learning*, zkr. ML) se zabývá možnostmi počítače „učit se“ ze vstupů, které mu poskytneme. Cílem strojového učení je získat určitou znalost (obvykle v podobě počítačového programu) na základě vstupních, trénovacích dat. Není překvapením, že v závislosti na typu a komplexnosti vstupů, požadovaného výstupu a povaze úlohy, existují různé algoritmy učení a typy strojového učení. Hlavním cílem strojového učení je zobecnění vstupních trénovacích dat a vytvoření obecného matematického modelu, který co nejpřesněji určí správný výsledek v nových případech.

Předtím, než se podíváme na základní rozdělení algoritmů učení, zamysleme se, v jakých situacích může být strojové učení užitečné. Podle [29] existují dva obecné případy, kdy není efektivní nebo vůbec možné navrhnout algoritmus, který by přímo řešil požadovaný problém.

- **Komplexní problémy**

  Do této kategorie patří úlohy, které lidé vykonávají automaticky, ale neexistuje dostatečně podrobný popis, jak jsou tyto úlohy mozkem prováděny. Nemohou být tedy jednoduše algoritmizovány. Jedná se například o řízení auta, porozumnění mluvenému slovu nebo rozpoznávání objektů na obrázku.

  Druhá skupina úloh, která využívá metod strojového učení, souvisí s analýzou velkých a komplexních množin dat. S přibývajícím množstvím dat je pro člověka téměř nemožné nové informace zpracovávat a vyhodnocovat. Jedná se například o medicínská nebo meteorologická data, vyhledávání a vyhodnocování obsahu internetových stránek a ostatních rozsáhlých databází.

- **Potřeba přizpůsobování**

  Existuje mnoho úloh, které se mění s časem nebo podle potřeb uživatelů. U běžného programu to vede k potřebě úprav, přizpůsobování nebo personifikace pro konkrétní uživatele. Programy využívající strojové učení se přizpůsobí jejich vstupním datům a tím řeší tyto potřeby bez nutnosti úprav samotného kódu programu. Příkladem může být spamový filtr, který se postupně učí novým typům nevyžádané pošty.

### 3.1.1 Typy učení

Jak již bylo zmíněno v úvodu této kapitoly, učení je velmi široký pojem a na základě povahy řešeného problému existuje několik typů algoritmů učení. Uveďme základní rozdělení podle [29]:

- **Učení s učitelem**

  Vstupní data algoritmů učení s učitelem (*supervised learning*) obsahují informaci o požadovaném výstupu. Vstupní množina dat se nazývá trénovací množina. Každý prvek z této množiny obsahuje jeden nebo více vstupů a také požadovaný výstup. Algoritmus učení zohledňuje požadovaný výstup při trénování matematického modelu. Jak uvidíme později v textu, pro naučení neuronové sítě, implementované v naší aplikaci, byla použita právě tato metoda učení.

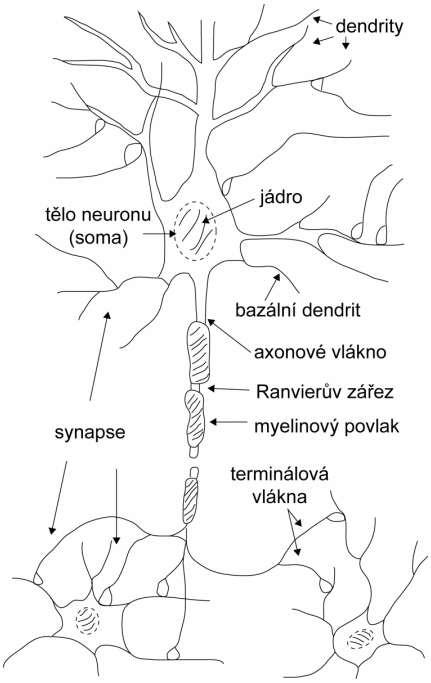
- **Učení bez učitele**

  Algoritmy učení bez učitele (*unsupervised learning*) mají rozdílnou trénovací množinu. Oproti algoritmům učení s učitel neobsahují prvky této množiny požadovaný výstup, který by mohl být využit při trénování. Při učení bez učitele jde obecně o hledání podobností vstupních dat, na základě kterých mohou být data kategorizována nebo rozdělena do určitých skupin, shluků.

- **Inkrementální a dávkové učení**

  Nezávisle na faktu, zda je či není při učení využit učitel, můžeme rozdělit algoritmy učení podle dostupnosti vstupních dat na začátku a během učení. Během inkrementálního učení (*online learning*) jsou vstupy zpracovávány postupně (inkrementálně) a také model je postupně upravován. Při dávkovém učení (*batch learning*) je výsledný model získán až po zpracování celé vstupní trénovací množiny (dávky).

Poznamenejme, že učení s učitelem a bez učitele lze kombinovat tak, že u části dat je známý požadovaný výsledek a u části dat není uveden. Tento typ učení se anglicky nazývá *semi-supervised learning*.




Obrázek 3: Schéma biologického neuronu. Převzato z [11].

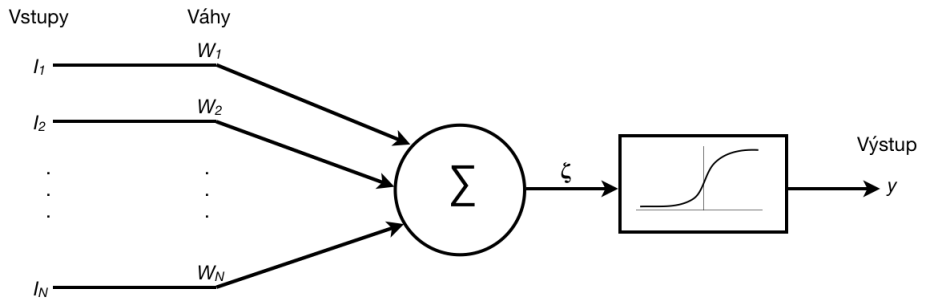
## 3.2 Umělé neuronové sítě

Umělá neuronová síť (*artificial neural network*, zkr. ANN) je matematický model využívaný v algoritmech strojového učení. Umělé neuronové sítě se snaží o simulaci lidského mozku, který je tvořen sítí neuronů. Modely simulující toto chování jsou velmi zjednodušené oproti skutečné biologické neuronové síti. V této kapitole uvedeme popis umělých neuronových sítí a princip jejich učení. Dále v textu potom uvidíme využití neuronové sítě ve vytvářené webové aplikaci. Informace v této kapitole jsou převzaty z [10], [11] a [19].

### 3.2.1 Neuron

Základním prvkem lidského nervového systému je neuron. Neuron je buňka, která přijímá a vysílá elektrické signály. V lidském mozku je neuronů velké množství. Sousední neurony jsou vzájemně propojeny pomocí synapsí. Synapse slouží k předávání vzruchů a tím k šíření nervového signálu. Synapse mohou mít budící (excitační) nebo tlumící (inhibiční) vlastnost. Podle této vlastnosti buď posilují nebo zmírňují šířený signál. Síť těchto propojených neuronů se nazývá neuronová síť. Biologické i umělé neurony přijímají vstupy z různých zdrojů (okolních neuronů) a poskytují jeden výstup.

Na obrázku 3 můžeme vidět schéma biologického neuronu. Z pohledu pochopení návaznosti na matematický model umělého neuronu jsou nejpodstatnější dendrity, kterými je signál šířen do těla neuronu a poté axon, který poskytuje




Obrázek 4: Schéma perceptronu.

výstup neuronu. Na obrázku můžeme vidět i schématické propojení více neuronů pomocí synapse.

### 3.2.2 Matematický model neuronu

Jednoduchý model neuronu se nazývá perceptron. V textu dále budeme pod pojmen neuron chápat umělý neuron, nikoli ten biologický. Na schématu 4 jsou zobrazeny základní částí perceptronu. Jak již bylo zmíněno, neuron má několik vstupů označených $I_1$, $I_2$, ..., $I_N$ a jeden výstup $y$. Váhy $W_1$, $W_2$, ..., $W_N$ simulují vlastnost excitace nebo inhibice synapsí.

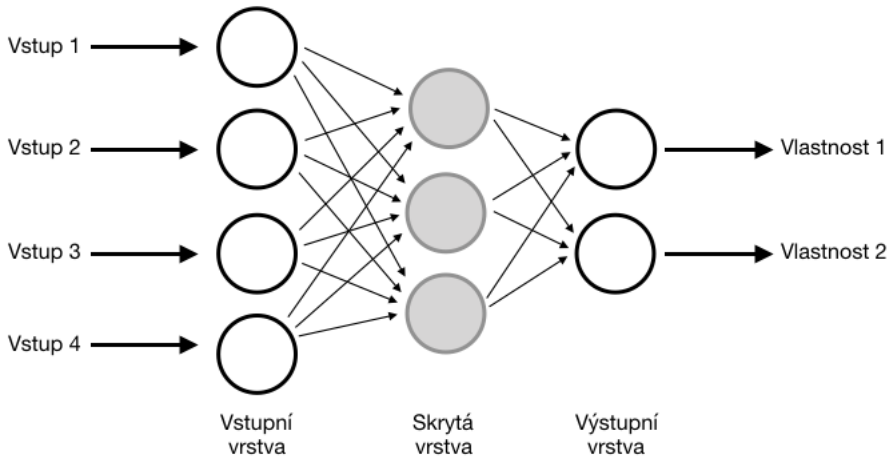
Vnitřní potenciál neuronu $\zeta$ je vypočítán jako součet všech součinů vektoru vstupů s vektorem vah:

$$\zeta = \sum_{i=1}^{N} I_i \cdot W_i \tag{7}$$

Pro výpočet výstupu $y$ neuronu se používá tzv. aktivační přenosová funkce $f$. Pro ilustraci uveďme nejjednodušší skokovou funkci. Všimněme si, že výsledná hodnota aktivační přenosové funkce $f$ z rovnice 8 závisí na velikosti prahové hodnoty (*thresholdu*) $\theta$. Pro úplnost uveďme, že přenosová funkce $f$ je obvykle nelineární a u vícevrstvých sítí se často využívá tzv. sigmoidální funkce nebo hyperbolický tangents (také podrobněji v [11]).

$$y = f(\zeta) = \begin{cases} 1, & f(\zeta) \geq \theta, \\ 0, & \text{jinak.} \end{cases} \tag{8}$$

Při učení nám jde tedy o určení hodnot vektoru vah a prahové hodnoty $\theta$ na základě trénovací množiny. Úpravou těchto parametrů upravujeme chování neuronu. Říkáme, že se neuron naučil trénovací množinu, pokud je pro každý vstupní vektor výsledek aktivační fuknce shodný s požadovaným výstupem. Poznamenejme, že takto popsaný neuron se dokáže naučit pouze lineárně separabilní trénovací množiny. Pokud tedy není trénovací množina lineárně separabilní, nelze




Obrázek 5: Schéma vícevrstvé sítě.

správně určit hodnoty parametrů vah a prahové hodnoty $\theta$. Řešením tohoto problému je vytvoření sítě perceptronů, kdy jsou výstupy neuronů vstupem jiných.

### 3.2.3 Vícevrstvé sítě

V předchozí kapitole jsme popsali základní model neuronu. Propojení perceptronů může tvořit jednovrstvou nebo vícevrstvou síť. V případě jednovsrtvé sítě je využíván název jednovrstvý perceptron, protože se ve skutečnosti jedná o několik nezávislých, ale paralelně pracujících neuronů. Jednovrstvý neuron je tvořen vstupní vrstvou a výstupní vrstvou neuronů. Vstupní vrstva je propojena se všemi neurony vrstvy výstupní. Vstupní vrstva není tvořena neurony, ale poskytuje pouze vstupní hodnoty pro vrstvu výstupní.

Pokud rozšíříme síť minimálně o jednu skrytou vrstvu, která se nachází mezi vstupní a výstupní vrstvou, získáme tzv. vícevrstvý perceptron. Musí navíc platit, že všechny neurony jsou propojeny se všemi neurony sousední vrstvy a nejsou propojeny mezi sebou v rámci jedné vrstvy tak, jak znázorňuje obrázek 5. Počet vstupních, výstupních i skrytých neuronů může být libovolný a je obvykle volen podle povahy řešeného problému. Signál je šířen ze vstupní vrstvy, přes skryté vrstvy k vrstvě výstupní, tedy jedním směrem. Z tohoto důvodu se takto definované sítě nazývájí sítě s dopředným šířením signálu a jsou nejčastěji používanými neuronovými sítěmi.

### 3.2.4 Algoritmus učení

Pro vícevrstvé neuronové sítě se problém učení rozšiřuje na problém nalezení vhodné topologie sítě a určení funkce, která je reprezentovaná sítí. Funkce reprezentovaná sítí musí co nejvíce odpovídat trénovací množině. Topologii sítě

je vhodné volit podle povahy řešeného problému, jak již bylo zmíněno. Algoritmus učení musí pro danou topologii najít parametry aktivační přenosové funkce.

Na příkladu uveďme sigmoidální funkci, která se u vícevrstvých sítí s dopředným šířením často využívá.

$$y = \frac{1}{1 + e^{-\lambda \cdot z}} \tag{9}$$

Kde $\lambda$ značí strmost funkce a $z$ je vyjádřeno následovně:

$$z = \sum_{i=1}^{N} I_i \cdot W_i - \theta \tag{10}$$

Jde nám tedy o nalezení hodnot prahu $\theta$, strmosti $\alpha$ a vah $W$ pro každou vrstvu sítě a pro každý neuron v ní. Učení je obvykle iterativní proces. V prvním kroku jsou hledané hodnoty parametrů ručně nastaveny, vypočítána hodnota $y$ a určena velikost odchylky $E$ od požadovaného výstupu. V dalším kroku probíhá úprava hodnot prahu $\theta$, strmosti $\alpha$ a vah $W$, dokud není dosaženo požadované přesnosti $\epsilon$ nebo není dosažen předepsaný maximální počet opakování.

Velmi známý algoritmus učení vícevrstvé neuronové sítě se nazývá *Backpropagation* (zkr. BP), který pro minimalizaci chyby využívá gradientní metodu. Rovnice gradientního sestupu lze interpretovat jako zpětné šíření chyby. Z této interpretace vychází název algoritmu *Backpropagation*. Poznamenejme, že jsme zde uvedli pouze obecný a zjednodušený popis učení neuronové sítě. Podrobnější informace včetně odvození rovnic algoritmu *Backpropagation* lze nalézt například ve čtvrté kapitole [19].

# 4 Metody zpracování obrazu

Zrak je nejdůležitější smysl člověka, kterým vnímá přibližně 80% informací [30]. Není tedy překvapením, že reprezentace a zpracování obrazu je velmi důležité a nachází své uplatnění v téměř všech oborech.

Digitální obraz lze chápat jako dvourozměrnou funkci $f(x, y)$, kde $x$ a $y$ jsou souřadnice, kterým je přiřazována intenzita barvy v tomto bodě. Pojmem digitální obraz budeme dále chápat vždy obraz rastrový. Vektorové obrazy, které jsou definované pomocí geometrických objektů nebudeme v textu využívat ani dále popisovat. Digitální obraz je složen z konečného počtu pixelů, které mají svou polohu a hodnotu. Při práci s obrazem je pracováno s jednotlivými pixely, jejich okolím nebo obecnější vlastností (části) obrazu. Zpracováním obrazu nazýváme operaci, která je použita na vstupní obraz a výstupem je podle typu operaci buď jiný obraz nebo například popis charakteristických vlastností vstupního obrazu.

V textu se zaměříme pouze na vybrané typy operací, které byly využity při návrhu a implementaci algoritmu pro rozpoznávání fotografií čolka. Informace v této kapitole byly čerpány z [7].

## 4.1 Segmentace

Segmentace obrazu je skupina metod zpracování obrazu určená k rozdělení pixelů vstupního obrazu do několika skupin (segmentů), jak už sám název napovídá. Cílem segmentace je upravit, většinou zjednodušit vstupní data tak, aby byla dále lépe analyzovatelná. Výsledkem segmentace jsou skupiny pixelů, které mají shodnou vlastnost. Jedná se například o jejich barvu nebo intenzitu. Nejčastěji je segmentace využívána pro nalezení objektů v obraze, oddělení objektů od pozadí obrazu a jejich rozpoznávání.

## 4.2 Prahování

Jednou ze základních metod segmentace je metoda prahování (*thresholding*). Výsledný obraz daný funkcí $g(x, y)$ je definovaný jako:

$$g(x,y) = \begin{cases} 1, & f(x,y) > T, \\ 0, & f(x,y) \leq T. \end{cases} \tag{11}$$

Kde $T$ je hodnota prahu a $f(x, y)$ je hodnota pixelu na souřadnicích $x$ a $y$ vstupního obrazu. Poznamenejme, že se jedná o normalizovaný typ zápisu rovnice a v praxi se využívají pro hodnoty pixelů i prahu hodnoty z intervalu $[0, 255]$.

Podle typu prahu rozdělujeme prahování na globální a lokální. V případě globálního prahování je $T$ konstanta a práh je tedy stejný pro každý pixel vstupního obrazu. U lokálního prahování se hodnota $T$ liší v závislosti na souřadnicích $x$ a $y$ vstupního obrazu. Jednotlivé metody prahování se liší ve způsobu výběru a případných úpravách prahu.

Obrázek 6: Původní fotografie (vlevo) a otsuova metoda prahování aplikovaná na červený kanál (vpravo).

### 4.2.1 Otsuova metoda

Základní metody globálního prahování pracují s pevně danou hodnotou prahu. Vhodná hodnota se většinou u těchto metod zjišťuje testováním a minimalizací chyby v závislosti na výsledném obrazu. Obecně je velmi obtížné určit konstantní hodnotu pro různé vstupní fotografie. Otsuova metoda prahování je pokročilejší metoda globálního prahování. Narozdíl od jednodušších metod není hodnota prahu vybrána libovolně, ale je vypočítána na základě histogramu obrazu.

Histogram obrazu je diskrétní funkce $h(r_k) = n_k$, kde $r_k$ je hodnota intenzity (obvykle z intervalu $[0, 255]$) a $n_k$ je počet počet pixelů s intezitou $r_k$. Pro Otsuovu metodu prahování je nejvhodnější takový obraz, jehož histogram má dva vrcholy. Prahem je zvolena hodnota právě mezi těmito vrcholy. Pokud takovou vlastnost vstupní obraz nemá, prahování nebude přesné.

Nalezení prahové hodnoty je založeno na výpočtu tzv. variace v rámci třídy (*Within-Class Variance*) pro každou hodnotu intenzity. Prahem je právě ta intenzita, pro kterou je hodnota z rovnice (12) minimální. Převzato z [32].

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{12}$$

Hodnoty $q$ (váhy) a $\sigma$ (variace) jsou vypočítány následovně:

$$q_1(t) = \sum_{i=1}^{t} H(i), \qquad q_2(t) = \sum_{i=t+1}^{I} H(i), \tag{13}$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iH(i)}{q_1(t)}, \qquad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iH(i)}{q_2(t)}, \tag{14}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{H(i)}{q_1(t)}, \qquad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{H(i)}{q_2(t)}. \tag{15}$$

Z barevné fotografie je pro prahování otsuovou metodou v implementaci využit červený kanál, jehož histogram nejlépe rozděluje tmavé části těla zvířete od jeho těla a od pozadí fotografie. Z obrázku 6 je patrné, že samotné využití této metody není dostatečné a proto je otsuova metoda použita společně s kombinací dalších metod prahování, které jsou popsány dále.

### 4.2.2 Adaptivní prahování

Adaptivní, někdy nazývané také jako lokální nebo dynamické prahování nevyužívá konstantní hodnotu prahu. Hodnota prahu je dynamicky měněna v závislosti na poloze a často i okolí pixelu. Pokud jsou například světelné podmínky na různých místech fotografie rozdílné, proměnlivá hodnota prahu může zajistit požadovaný výsledek. V těchto situacích není obvykle globální prahování dostatečné.

V implementaci jsou využity dvě metody adaptivního prahování. První z nich počítá lokální práh jako průměrnou hodnotu (*Mean Thresholding*) intenzit zadaného okolí. Na vstupu má metoda zadanou velikost okolí, které má při výpočtu zohledňovat. Pro každý pixel je tedy nejdříve určena hodnota prahu jako průměrná hodnota všech intenzit pixelů v zadaném okolí a následně je výsledek získán výpočtem rovnice (11) pro prahování.

Druhá využitá adaptivní metoda využívá k výpočtu prahu vážený součet hodnot intezit okolních bodů. Pro určení vah je použito Gaussovo rozdělení pravděpodobnosti. Často se pro toto rozdělení používá název normální rozdělení pravděpodobnosti. Toto rozdělení se mimo jiné využívá v přírodních vědách k reprezentaci hodnoty náhodné proměnné, u které neznáme její rozdělení.

### 4.2.3 Kombinace metod

Z příkladů aplikace jednotlivých metod prahování zobrazených na obrázcích 6, 7 a 8 je patrné, že volba metody záleží na potřebném výsledku. Obecnou výhodou výše popsaných způsobů prahování je, že není potřebné určit hodnotu prahu na vstupu. Každá z metod má tedy na vstupu pouze dvourozměrné pole hodnot v intervalu $[0, 1]$, respektive $[0, 255]$ a v případě adaptivního prahování je určena velikost okolí. Výstupem prahování je opět dvourozměrné pole, tentokrát binárních hodnot 0 a 1, respektive 0 a 255. Podařilo se nám tedy segmentovat interval hodnot na pouhé dvě hodnoty. Hodnota nula je na obrázku černá a jedna bílá.

Poznamenejme, že pro naplnění požadavku vstupních hodnot v intervalu $[0, 1]$ musí být barevná fotografie nejdříve převedena do odstínů šedi. Tato operace je standardně prováděna váženým součtem jednotlivých RGB kanálů. V grafické

Obrázek 7: Původní fotografie (vlevo) a *Mean Thresholding* aplikovaný na červený kanál (vpravo).

Obrázek 8: Původní fotografie (vlevo) a gaussova metoda prahování aplikována na červený kanál (vpravo).

Obrázek 9: Původní fotografie (vlevo) a výsledek kombinace tří metod prahování (vpravo).

knihovně OpenCV, která je popsána v kapitole 6.2 je implementace této operace definována dle [31] následovně:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{16}$$

Pro každý vstupní pixel, tedy trojici hodnot je vypočítána hodnota $Y$, která splňuje potřebnou vlastnost.

Při vývoji bylo experimentálně zjištěno, že největší rozdíly na fotografiích čolků se vyskytují v červeném (R) kanálu. Při využití pouze této barvy dochází k lepší segmentaci pro další zpracování. Převod na odstíny šedi se oproti rovnici (16) zjednodušuje na:

$$Y = R \tag{17}$$

Jak již bylo zmíněno, výsledkem prahování je dvourozměrné pole hodnot nula a jedna. Pro získání otpimálního výsledku pro další zpracování, tedy určení skvrn na těle čolka, byly uvedené tři metody zkombinovány. Obecně lze dle provedených experimentů říci, že otsuova metoda prahování ze vstupní fotografie segmentuje tělo bez okolního šumu. Adaptivní metody prahování zaručují lepší přesnost v segmentaci skvrn, ale z principu výpočtu lokálního prahu je přítomný i šum, respektive nežádoucí oblasti černé barvy v okolí těla.

Kombinací obrazů je možné provádět díky běžným aritmetickým operacím, které lze na obrazy aplikovat. Pro účely této práce byla využita operace sčítání

v tzv. saturované podobě. Požadovaný výsledek je vždy ze stejného intervalu jako vstup. Operaci lze zapsat následovně:

$$Y_1 + Y_2 = \min(\max(Y_1 + Y_2, 0), 1) \tag{18}$$

Pro hodnoty nula nebo jedna na vstupu bude tedy výsledek vždy také nula nebo jedna. V grafickém vyjádření získáme černou barvu pouze tam, kde byla původně černá na všech sčítaných obrázcích. Tímto způsobem dosáhneme toho, že jsou odstraněné nežádoucí oblasti černé barvy v okolí těla a zároveň jsou skvrny na těle dostatečně přesné pro porovnávání. Výsledek rozebíraného příkladu je uveden na obrázku 9.

## 4.3 Porovnávání fotografií

Pro porovnávání fotografií se využívají již upravené fotografie metodami uvedenými výše. Na vstupu máme tedy dva binární obrazy a potřebujeme získat míru jejich podobnosti, respektive rozdílnosti. K výpočtu využíváme tzv. momenty obrazu, které jistým způsobem popisují objekty na fotografii. Existuje několik typů těchto momentů, jak je uvedeno v [6]. Konkrétně pro srovnání jsou ideální tzv. invarianty momentů, které jsou neměnné při posunu, rotaci i škálování daného objektu. Tyto invarianty se také anglicky označují *Hu moment invariants*. Pro jejich efektivní výpočet se v OpenCV využívá funkce HuMoments, která dle vývojářské dokumentace [33], počítá invarianty následujícím způsobem:

$$hu[0] = \eta_{20} + \eta_{02} \tag{19}$$

$$hu[2] = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{20}$$

$$hu[3] = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{21}$$

$$\begin{aligned} hu[4] = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ +(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{22}$$

$$hu[5] = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{23}$$

$$\begin{aligned} hu[6] = (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ -(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{24}$$

kde $\eta_{ji}$ je normalizovaný cen trální moment počítaný funkcí moments. Rozdíl mezi obrazy lze vypočítat s využitím výše uvedených invariantů. OpenCV implementuje následující tři metody výpočtu rozdílu mezi objekty $A$ a $B$:

$$D_1(A, B) = \sum_{i=1\ldots7} \left| \frac{1}{m_i^A} - \frac{1}{m_i^B} \right| \tag{25}$$

$$D_2(A,B) = \sum_{i=1\ldots7} \left| m_i^A - m_i^B \right| \tag{26}$$

$$D_3(A,B) = \max_{i=1\ldots7} \frac{\left| m_i^A - m_i^B \right|}{|m_i^A|} \tag{27}$$

kde $m_i^A = \text{sign}(h_i^A) \cdot \log h_i^A$ a $m_i^B = \text{sign}(h_i^B) \cdot \log h_i^B$ pro momenty $h_i^A$, $h_i^B$ objektů $A$ a $B$.

Výsledkem je tedy rozdíl vzdálenosti $D$. Čím menší je tato vzdálenost, tím více jsou objekty podobné. Stejné objekty budou mít nulovou vzdálenost. Aplikace využívá takto popsaný způsob výpočtu vzdálenosti na všechny objekty, zejména tedy na skvrny na těle čolka získané pomocí segmentace, které se nachází na fotografii. Na základě provedených experimentů se ve finální implementaci využívá výpočet uvedený v rovnici 26, který dosahuje na fotografiích čolků nejlepších výsledků.

# 5 Uživatelská příručka

V následující kapitole rozvedeme možnosti aplikace a porovnáme je s nalezeným existujícím řešením, kterým je německý program *AmphIdent*.[2] Podrobně se také budeme věnovat jednotlivým částem aplikace z uživatelského pohledu.

## 5.1 Účel aplikace

Aplikace byla navržena a vyvinuta přímo ve spolupráci s cílovými uživateli. Jedná se o pracovníky katedry ekologie a životního prostředí na Přírodovědecké fakultě Univerzity Palackého v Olomouci. Při návrhu aplikace bylo možné využít zkušeností se současným postupem uplatňovaným při odhadu velikosti populace.

Jak již bylo zmíněno ve 2 kapitole, proces lze rozdělit na několik kroků. Nejdříve je potřeba získat vstupní data, tedy odchytit jedince čolků a pořídit jejich fotografie. Po tomto kroku již není nutné provádět žádné další operace a každá fotografie čolka může být nahrána a analyzována v aplikaci.

Aplikace poskytuje přehledné rozčlenění fotografií podle lokality, data odchytu a pohlaví jedince zachyceného na fotografii. Při nahrávání je fotografie poprvé analyzována, a je určena orientační kvalita nahráváné fotografie a také pohlaví čolka. Po uložení je zahájena detailnější analýza zahrnující segmentaci skvrn, nalezení kontur a porovnání s ostatními fotografiemi. Na detailní stránce, která je vytvořena pro každého jedince, jsou kromě lokality a data ochytu zobrazeni také nejpodobnější čolci. Potvrzení nalezené shody je ponecháno na uživatele. Po potvrzení je čolek označen jako znovu odchycený (*recaptured*) a tento fakt je následně zohledněn při výpočtu odhadu velikosti populace.

Pokud porovnáme popsané možnosti aplikace s aktuálním manuálním postupem vidíme, že aplikace poskytuje přehledné rozdělení všech dat, kompletní potřebnou analýzu a také výpočet odhadu velikosti populace. Pokrývá tedy časově nejnáročnější procesy při zkoumání populace čolků. Aplikace lze v budoucnu rozšířit o propojení například s mobilní aplikaci přes kterou budou pořízovány fotografie čolků.

## 5.2 Existující řešení

Před implementací popisovaného řešení byla testována aplikace *AmphIdent*, která má podobný účel. Aplikace slouží k nalezení podobných vzorů na fotografiích a k jejich vzájemnému porovnávání. Je možné porovnávat více druhů. Dle oficiálního webu [1] vy měla aktuální verze programu umožňovat porovnávání fotografií čolka velkého (*Triturus cristatus*), kuňky obecné (*Bombina bombina*) a žlutobřiché (*Bombina variegeta*), mloka skvrnitého (*Salamandra salamandra*) a axolotla mramorového (*Ambystoma opacum*).

Jedná se o desktopovou aplikaci, která je určena pouze pro operační systém Windows. Nevýhodou je tedy závislost na dané platformě a při testování se pro-

[2] Dostupný na webu http://www.amphident.de.

gram na několika počítačích ani nepodařilo spustit. Jako další nevýhoda může být chápána poměrně vysoká cena licence programu, který je poskytován ve formě modulů, jejichž každá část je placena zvlášť. Program je zaměřený pouze na porovnávání samotných fotografií a nedisponuje proto ostatními požadovanými vlastnosti. Jiný podobný program nebo aplikace nebyly nalezeny. Z těchto důvodů bylo navrženo kompletně vlastní řešení.

## 5.3 Práce s aplikací

Aplikace je podporována ve všech aktuálních webových prohlížečích. V operačním systému Windows však doporučujeme nahradit Internet Explorer 11 prohlížečem Google Chrome. Pro plnou funkcionalitu je nutné mít v prohlížeči aktivovanou podporu JavaScriptu. Odkaz na aktuální verzi aplikace dostupnou z internetu je uveden v přiloženém souboru na CD (příloha A).

V horním menu je vidět rozdělení aplikace na několik částí. Na úvodní stránce je přehled všech lokalit. Lokalita je zde chápána jako místo, na němž se vyskytují a jsou odchytáváni čolci. V každé lokalitě je obvykle prováděno několik odchytů čolků a to v různých dnech. Nejdříve je tedy v aplikaci nutné vytvořit lokalitu (pokud již požadovaná lokalita neexistuje) a následně upřesnit podrobnosti pro konkrétní den odchytu. Poté je možné nahrávat, kategorizovat a porovnávat fotografie čolků. Velikost populace se vztahuje vždy k určité lokalitě a je počítána na základě údajů z pořízených fotografií v jednotlivých odchytech dané lokality.

### 5.3.1 Lokalita

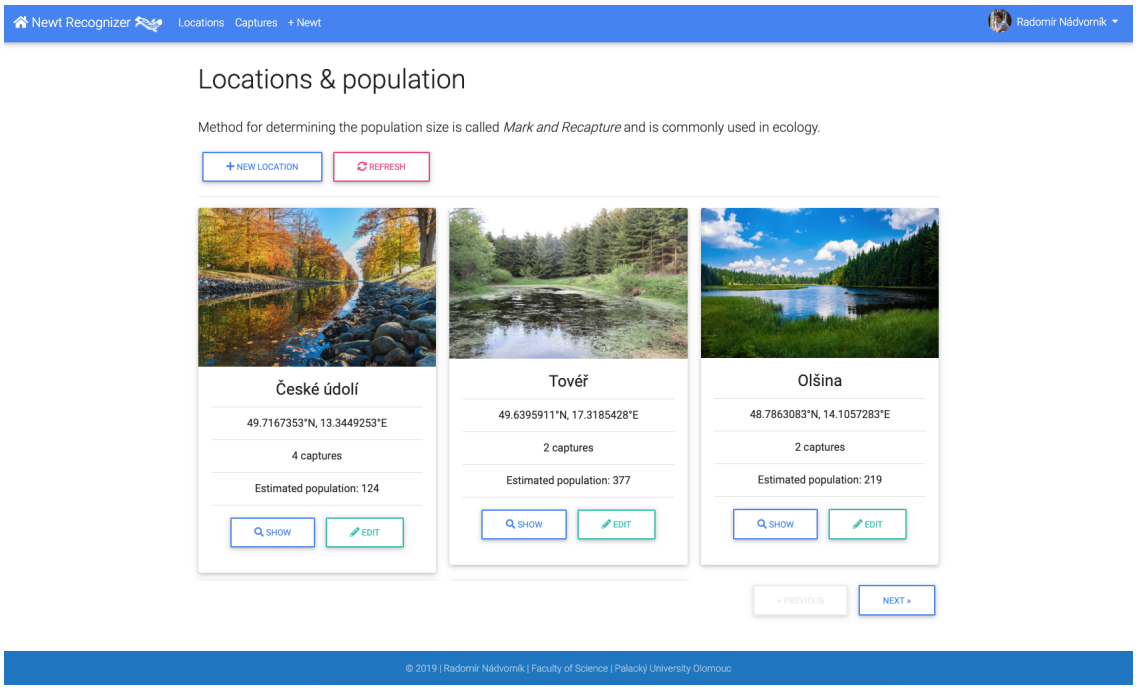
Při vytváření lokality vloží uživatel její název a polohu. Volitelně je možné přidat nadmořskou výšku a náhledový obrázek pro lepší přehlednost. Poloha se zadává ve tvaru GPS souřadnic. Pro zjednodušení je možné zvolit oblast na zobrazené mapě. Po kliknutí do mapy se automaticky doplní zeměpisná šířka, délka i nadmořská výška. Po uložení se lokalita zobrazí v seznamu všech doposud vytvořených.

V přehledu lokalit jsou jednotlivá místa seřazená podle počtu odchytů v dané oblasti a to od největšího. U každé oblasti jsou uvedné informace vložené na začátku a dále také počet odchytů a velikost odhadované populace. Po kliknutí na obrázek nebo název lokality se uživatel dostane na přehled všech odchytů ve zvolené lokalitě. Viz obrázek č. 10.

### 5.3.2 Odchyt

Jak již bylo řečeno, k odchytu čolků dochází v jedné lokalitě opakovaně. Jednotlivé odchyty jsou odlišeny datem. Pro vytvoření odchytu je tedy povinné zvolit lokalitu a vyplnit datum. Vhodné je doplnit tyto informace o jméno autora a název organizace, která odchyt prováděla.

Odchyty jsou přehledně zobrazeny v tabulce a seřazeny od nejnovějšího podle data odchytu. V přehledu jsou kromě na počátku zadaných informací také počty




Obrázek 10: Ukázka uživatelského rozhraní seznamu lokalit.

čolků v každém z odchytů. Je zde uveden počet samců a samic čolků, celkový počet a také počet znovu chycených jedinců. Při kliknutí na řádek určitého odchytu se uživateli zobrazí přehled všech čolků vyfocených v den zvoleného odchytu.

### 5.3.3 Přehled čolků

Po zobrazení přehledu čolků se uživateli zobrazí odlišná tabulka, která umožňuje vyhledávání a řazení záznamů. Uživatel může také zvolit počet záznamů zobrazených najednou. Užitečná je i navigační lišta nad tabulkou, která umožňuje návrat na přehled všech odchytů v dané lokalitě nebo na úvodní stránku. V tabulce jsou u každého čolka uvedeny nejdůležitější údaje pro orientaci a hledání konkrétního záznamu. Po kliknutí na vybraný záznam se zobrazí detailní stránka čolka, která je popsána v kapitole 5.3.5.

### 5.3.4 Přidání čolka

Přidání nového čolka je zahájeno nahráním jeho fotografie. Soubor je možné nahrát výběrem z počítače nebo přetažením do znázorněné oblasti. Při nahrávání dojde k analýze obrazu pomocí neuronové sítě a uživateli se zobrazí odhadovaná kvalita nahrané fotografie. Jako dostatečně kvalitní je chápán obrázek, na kterém byl rozpoznán čolek s pravděpodobností alespoň 60%. Tato hodnota je spíše orientační a zároveň slouží jako upozornění pro uživatele, aby v případě nízkého výsledku nahrál fotografii jinou. Analýza se pokusí určit také oblast, na které se

nachází tělo čolka. Takto zjištěná oblast slouží jako výchozí nastavení pro ořez fotografie. Uživatel může oblast případně upravit, fotografii ořezat a uložit.

Po uložení se uživateli zobrazí nová stránka pro upřesnění detailů k právě nahranému záznamu. Při ukládání dochází k určení odhadu pohlaví čolka pomocí další neuronové sítě a uživateli tak stačí pouze zkontrolovat tento odhad. Datum a místo odchytu je také automaticky předvyplněno. Uživatel jej může samozřejmě změnit. Na základě zvoleného místa, data odchytu a pohlaví je vytvořeno pojmenování daného čolka. Tvar pojmenování vychází z praxe v tomto oboru. Automaticky vyplněné pojmenování lze také libovolně měnit. Pokud je uživatel do aplikace přihlášen, doplní se jeho jméno do pole autora záznamu. V opačném případě jej musí vyplnit manuálně. Zbývající pole nejsou povinná. Jedná se o tzv. *SVL* představující délku těla zvířete v milimetrech a o pole pro další doplňující informace. Po uložení vyplněného formuláře je záznam zařazen podle zvolených hodnot do odpovídající lokality a odchytu. Zároveň je na pozadí zahájena analýza a porovnávání s ostaními fotografiemi čolků. Návrhy podobných fotografií se zobrazují na detailní stránce, kde je uživatel může potvrdit nebo smazat.

### 5.3.5 Detail čolka

Výběrem konkrétního řádku v přehledu čolků jsou uživateli zobrazeny detaily zvoleného čolka. Pod informacemi, které byly zadány při nahrávání fotografie, jsou tabulky s navrženými a potvrzenými shodnými obrázky. V seznamu nalezených podobností jsou zobrazeny nejpodobnější čolci, kteří jsou seřazeni od nejvyššího procenta podobnosti. Po potvrzení, že se jedná o stejného čolka, se záznam zobrazí v seznamu potvrzených shod. Z pohledu odhadu velikosti populace to znamená, že byl tento čolek již někdy odchycen.

Porovnávání se provádí pouze pro čolky stejného pohlaví chycené ve stejné lokalitě avšak v jiném odchytu, než čolek, pro kterého je zobrazen detail. V případě, že takový neexistuje nebo je vzájemná podobnost všech vyhovujících záznamů jen velmi malá, je seznam návrhů prázdný. Seznam návrhů může být také prázdný, pokud nebyla analýza zatím dokončena. Uživatel může tlačítkem manuálně zahájit nové porovnání odpovídajích záznamů, pokud je to potřeba. Nesprávné návrhy lze ze seznamu odstranit pro zachování přehlednosti. S využitím navigační listy v horní části stránky je snadné se vrátit zpět na přehled všech čolků v daném odchytu nebo na seznam všech odchytů v lokalitě čolka, jehož detail je zobrazen.

# 6 Využité technologie

V kapitole 3 a 4 jsou uvedeny metody strojového učení a metody zpracování obrazu využité v popisované aplikaci. Nejen tyto metody jsou implementované pomocí technologií, které právě stručně představíme. Aplikaci můžeme rozdělit na *frontend* a *backend*. Pro každou z těchto částí byla pochopitelně využita jiná technologie a nástroje. Předmětem práce je návrh i implementace obou těchto částí. Podrobněji je architektuře aplikace věnována programátroská dokumentace v kapitole 8.1. Pro nasazení aplikace a pro umožnění přístupu do aplikace uživatelům internetu je využívána platforma Google Cloud, kterou se zabývá kapitola 7.

## 6.1 Programovací jazyk Python

Serverová část aplikace je naprogramována v jazyce Python. Python je interpretovaný, objektově orientovaný programovací jazyk. Jeho předností je jednoduchá syntaxe a přenositelnost, která umožňuje spuštění programu na systémech Windows, macOS i Unix. Jednoduchost syntaxe vychází z rozsáhlé standardní knihovny funkcí, která je jeho součástí a dokáže řešit základní i pokročilejší problémy. Navíc je k dispozici také mnoho knihoven třetích stran, které jsou navrženy pro tento programovací jazyk. Podrobnější informace jsou k nalezení na oficiálních webových stránkách [25], z nichž jsou čerpány i výše uvedené údaje. Python byl zvolen i s ohledem na podporované programovací jazyky platformou Google Cloud a také kvůli možnosti využít funkce grafické knihovny OpenCV.

## 6.2 Grafická knihovna OpenCV

OpenCV je zkratka grafické knihovny otevřeného softwaru tzv. *Computer Vision*, volně přeložitelné jako počítačové zpracování obrazu. Užití této knihovny je zdarma pro akademické i komerční účely. Knihovna, dle oficiálního webu [23], podporuje Python, C++ a programovací jazyk Java na operačních systémech Windows, Linux, macOS, iOS a Android. Knihovna OpenCV je v aplikaci využita pro zpracování obrazu. Efektivně implementuje metody popsané v kapitole 4 používané v aplikaci. Přiklad zdrojového kódu, který využívá knihovnu OpenCV je uveden v kódu 1.

## 6.3 Framework Flask

Flask je webový framework, který vývojářům poskytuje sadu nástrojů a knihoven pro vývoj webových aplikací. Tento framework zahrnuje vývojový server, nástroje pro testování a ladění. Dokumentace je dostupná online z [5]. Flask je založený na nástrojích Werkzeug a Jinja. Werkzeug je komplexní WSGI (*Web Server Gateway Interface*) knihovna. WSGI specifikuje způsob zpracování požadavků a komunikace aplikace s webovým serverem. Jinja je dle [15] jeden

```
import cv2
import numpy as np
import urllib.request

def url_to_image(url):
    # convert downloaded image to a NumPy array
    # read it into OpenCV format
    resp = urllib.request.urlopen(url)
    img_np_array = np.asarray(bytearray(resp.read()), dtype="uint8")
    image = cv2.imdecode(img_np_array, cv2.IMREAD_GRAYSCALE)

    return image
```

Zdrojový kód 1: Příklad využití OpenCV knihovny.

z nejpoužívanějších šablonovacích nástrojů pro Python. Pomocí šablon je definováno základní rozložení jednotlivých stránek a také prvky, které jsou dynamické. Na příkladu 2 můžeme vidět příklad využití Jinja šablony pro zobrazení obrázku čolka. V šabloně je definována pozice a velikost obrázku, ale samotný obsah - v tomto případě zdroj obrázku a jeho popis, je dynamický. Tímto příkladem se také dostáváme k dalším využitým technologiím, kterými jsou HTML, CSS a JavaScript.

```
{% if newt.imageUrl %}
    <div class="card">
        <!-- Card image -->
        <div class="view overlay">
            <img class="card-img-top" src={{newt.imageUrl}}
                 alt="{{newt.label}}">
            <a href="{{newt.imageUrl}}">
                <div class="mask rgba-white-slight"></div>
            </a>
        </div>
    </div>
{% endif %}
```

Zdrojový kód 2: Příklad využití Jinja šablony.

## 6.4 HTML, CSS a JavaScript

Uživatel pro práci s aplikací využívá webový prohlížeč, který získáva data ze serveru. V tomto případě se jedná o kombinaci souborů využívajících HTML, CSS a JavaScript.

HTML (*Hyper Text Markup Language*) je standardní značkovací jazyk pro webové stránky a aplikace. Stránky jsou složeny z HTML bloků, které popisují

strukturu dokumentu. V minulosti byl pomocí HTML definován i samotný vzhled aplikace.

CSS (*Cascading Style Sheets*) neboli kaskádové styly nahradily HTML jazyk v oblasti formátování a vzhledu webu. Oddělení struktury a vzhledu stránky poskytuje větší flexibilitu ve všech ohledech. Díky CSS je docíleno větší jednoduchosti úprav díky nižšímu opakování stejného kódu. Oddělením formátováním od obsahu je také jednodušší různě zobrazovat stejnou stránku v odlišných situacích, například pro správné zobrazení na různých velikostech obrazovky. Podrobněji například v [3].

JavaSript je skriptovací jazyk. V aplikaci je využit jako rozšíření HTML stránek a jeho interpretaci tak zajišťuje webový prohlížeč. Díky JavaSriptu je možné například měnit a animovat jednotlivé prvky stránek nebo kontrolovat hodnoty odesílaných formulářů na straně klienta a mnoho dalšího. Pro úplnost je přiložen odkaz na oficiální web [14].

## 6.5 Framework Bootstrap

Pro vývoj frontendové části aplikace je využit framework Bootstrap [2] a jeho Material Design rozšíření [22]. Bootstrap je sada nástrojů a funkcí pro vývoj responzivních aplikací pomocí HTML, CSS a JavaScriptu. Tento framework je zdarma pro osobní i komerční účely, obsahuje přes 500 prvků uživatelského rozhraní a mnoho CSS animací. Bootstrap je kompabilitní se všemi moderními webovými prohlížeči a neustále se vyvíjí. Stejně jako u ostatních frameworků je vývojář aplikace postaven do role uživatele a zaměřuje se převážně na způsob využití jednotlivých komponentů bez potřeby jejich dokonalé znalosti.

Využití frameworku Bootstrap také pomáhá zachovat jednotný vzhled uživatelského rozhraní a ovládacích prvků napříč celou aplikací. Rozšíření pro Material Design [21] implementuje vizuální jazyk spojující osvědčené principy designu s inovací v technologiích a vědě. Tento jazyk je navržen společností Google a poprvé byl využit v uživatelském rozhraní operačního systému Android. Následovalo rozšíření napříč platformami a s tím spojené využívání ve webových prohlížečích.

# 7 Google Cloud

V kapitole 6 je uveden přehled využitých technologií v aplikaci. Záměrně byla vynechána podstatná část technologií využívaná napříč aplikací a jako infrastrukturní prvky pro hostování produkční verze aplikace. Jedná se o platformu Google Cloud, které se věnuje tato kapitola. Informace jsou čerpány z oficiálních webových stránek výrobce [8], zejména z dostupné dokumentace.

Platforma Google Cloud zahrnuje přes 100 produktů z různých oblastí informačních technologií. Google Cloud poskytuje produkty formou služby v oblastech umělé inteligence, serverové infrastruktury, databází, analýzy dat, vývojářských nástrojů, internetu věcí, nástrojů pro správce infrastrukury, sítí, bezpečnosti, úložiště a mnoho dalších. Dále se text zaměřuje zejména na ty produkty, které jsou využívány aplikací pro klasifikaci fotografií čolků. Nutno podotknout, že podobně rozsáhle portfolio služeb nabízí i Microsoft Azure[3] nebo Amazon Web Services.[4]

## 7.1 App Engine

Google App Engine je služba pro hostování aplikací a backendových částí aplikací bez potřeby spravovat servery a infrastrukturu, na které je aplikace nasazena. App Engine poskytuje vysokou míru automatického škálování výpočetního výkonu infrastruktury pro běh aplikace a je tak vhodná pro malé i rozsáhlé aplikace. Vývojáři se tedy nemusí zabývat správou a konfigurací serverů a mohou se více věnovat samotnému vývoji.

App Engine přímo podporuje aplikace v jazycích Java, PHP, Node.js, Python, C#, .Net, Ruby a Go, ale umožňuje i nasazení vlastních běhových prostředí a frameworků. Přímou součástí této služby je verzování aplikace, které umožňuje kdykoli spustit kteroukoli z přechozích verzí nebo řízení provozu, díky kterému je snadné jej rozdělit do více souběžně spuštěných verzí aplikace a postupně testovat například nové funkcionality. Pro případné řešení chyb je k dispozici detailní monitorovací a diagnostický nástroj, který je společný napříč všemi službami Google Cloud platformy.

## 7.2 Datastore a Storage

Google Cloud poskytuje několik možností uložení dat. V tomto případě je využíván Google Datastore a Cloud Storage. Google Datastore je NoSQL databáze, která automaticky zajišťuje replikaci a konzistenci dat a zajišťuje tak vysokou dostupnost s významnou mírou škálování výkonu, který se přizpůsobuje nárokům aplikace. Díky schématu databáze je méně náročné provádět změny ve struktuře dat při vývoji aplikace. Datastore samozřejmě umožňuje vyhledání dat na základě hodnot atributů a jejich řazení dle požadavků uživatele databáze.

[3] https://azure.microsoft.com/
[4] https://aws.amazon.com/

Google Cloud Storage poskytuje jednotné úložiště pro různé typy scénářů. V závislosti na potřebném výkonu a frekvenci přístupu lze volit a také měnit odpovídající typ třídy, do které jsou data ukládána. Stejně jako u ostatních služeb je garantovaná vysoká dostupnost dat a možnosti škálování. Aplikace využívá toto úložiště pro ukládáních všech fotografí čolků.

## 7.3 Cloud Functions

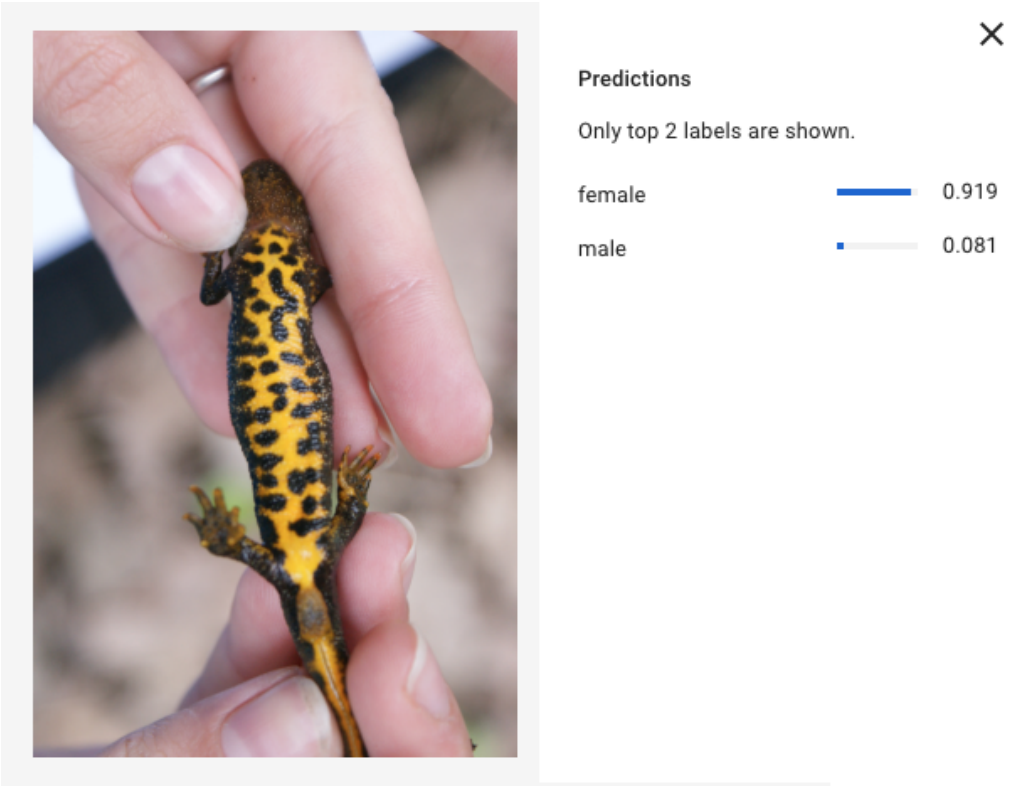
Cloud Functions (cloudové funkce) jsou navrženy k vykonávání jednoduchých programů na základě událostí. Slouží k propojení nebo rozšíření funkcionality cloudových služeb a aplikací. Na základě události v aplikaci nebo jiné cloudové službě dojde ke spuštění a vykonání kódu cloudové funkce, která může využívat další služby pomocí jejich aplikačních rozhraní a výsledek je zaslán zpět aplikaci. Takovou událostí může být například vytvoření nového souboru v úložišti, upravení nebo smazání entity v databázi a podobně. S využitím cloudových funkcí je aplikace složena z menších, vzájemně nezávislých částí. Aktualizace těchto funkcí není závislá na aplikaci, která je využívá. Pokud tedy potřebujeme upravit funkcionalitu, která je poskytována cloudovou funkcí, není nutné vytvářet další verzi aplikace. Cloudové funkce podporují programovací jazyky Node.js, Python a Go. V aplikaci jsou využívány například pro porovnávání fotografií a určování míry podobnosti.

## 7.4 Cloud Vision

Cloud Vision se řadí do služeb v oblasti umělé inteligence a strojového učení. V kapitole 3.2 jsou popsány neuronové sítě a princip učení. Cloud Vision poskytuje naučené neuronové sítě pro analýzu obrazu. Vývojářům jsou modely těchto sítí zpřístupněny pomocí aplikačního rozhraní (API). Rozhraní dokáže například detekovat a klasifikovat objekty nebo rozpoznat obličeje a dokonce i pocity lidí na fotografiích. V aplikaci je popsáno rozhraní využíváné pro detekci objektu a zjednodušuje tím ořezávání fotografie tak, aby tělo čolka zabíralo co největší část z ní. Zároveň je určena míra výskytu čolka na nahrané fotografii. Pokud je na fotografii čolek a fotografie je dostatečně kvalitní, ve výsledku analýzy se vyskytuje označení „čolek“ („True Salamanders And Newts“, „Smooth Newt“, „Newt“ a podobně) s vysokou pravděpodobností. Na základě výsledku této analýzy se může uživatel rozhodnout nahrát do aplikace jiný obrázek nebo pokračovat dále, jak je uvedeno v kapitole 5.3.4.

### 7.4.1 AutoML Vision

Dalším problémem, který lze řešit díky neuronovým sítím, je určování pohlaví čolka. Jedná se o velmi specifický požadavek, který vede k potřebě vytvoření vlastního modelu sítě. Cílem je tedy vytvořit takovou neuronovou síť, která na základě fotografie dokáže určit, zda se jedná o samce (*male*), nebo samici (*female*) čolka. Znaky, které jsou pro každé pohlaví specifické, jsme uvedli v kapitole 2.1.




Obrázek 11: Příklad určení pohlaví čolka velkého.

Z předchozích uskutečněných odchytů bylo získáno více než 2000 fotografií. Část fotografií byla vyřazena z důvodu špatné kvality. U zbývajících 1829 fotografií bylo určeno pohlaví čolka. Tím vznikly dvě skupiny, kde se na celkově 917 fotografiích vyskytují samice čolka velkého a na zbývajících 912 fotografiích samci.

Pro vytrénování sítě byly využity možnosti AutoML Vision pro klasifikování obrázků na základě námi poskytnutých údajů. Pro testovací množinu bylo náhodně zvoleno 170 obrázků z celkového počtu. Zbývající fotografie slouží k trénování sítě. Hranice (*threshold*) pro úspěšné určení pohlaví byla nastavena na 50%, což jednoduše vyplývá z faktu, že potřebujeme data rozdělit pouze na dvě skupiny. Pokud je tedy na fotografii například samice a odhad sítě je: „51% female“, je to považováno za dostatečné.

Současný model sítě dosahuje $84,7\%$ přesnosti (*precision*) a stejná hodnota platí také pro výtěžnost (*recall*). Přesnost je dle [4] definována jako procentuální poměr relevantních výsledků analýzy ke všem výsledkům získaných. Výtěžnost je poměr relevantních výsledků analýzy ke všem relevantním výsledkům ve zkoumaném vzorku bez ohledu na to, zda byly identifikovány. Na obrázku 11 je vidět konkrétní případ určení pohlaví čolka velkého na základě námi naučené neuronové sítě. Správně bylo vyhodnoceno, že se jedná o samici, která má světlejší barvu kloaky a také oranžový pruh na ocasu.

Aplikace využívá tuto neuronovou síť v části, kdy uživatel zadává podrobnosti

o nahrané fotografii. Na formuláři je předvyplněn údaj o pohlaví čolka na fotografii. Určení správného pohlaví je klíčové pro následnou analýzu a porovnávání s ostatními fotografiemi, protože jsou porovnávani právě pouze čolci stejného pohlaví.

# 8 Programátorská dokumentace

Předchozí kapitoly jsou věnovány aplikaci z uživatelského pohledu. Uvedli jsme také přehled metod zpracování obrazu, strojového učení a technologií, které byly využity při tvorbě webové aplikace. Následující text popisuje aplikaci z technického pohledu a poskytuje vazbu mezi dříve uvedenými informacemi a zdrojovým kódem aplikace. Zdrojový kód aplikace je přílohou této práce a v textu bude na jeho jednotlivé části odkazováno.

## 8.1 Architektura aplikace

Aplikaci lze rozdělit na tři základní vrstvy a to prezentační, aplikační a datovou. Prezentační vrstva zahrnuje uživatelské rozhraní umožňující uživateli (klientovi) komunikovat s ostatními vrstvami aplikace. Aplikační vrstva propojuje prezentační a datovou vrstvu. Aplikační část se někdy také nazývá částí logickou, protože zajišťuje funkcionalitu aplikace. Datová vrstva slouží k uložení a poskytování dat aplikace.

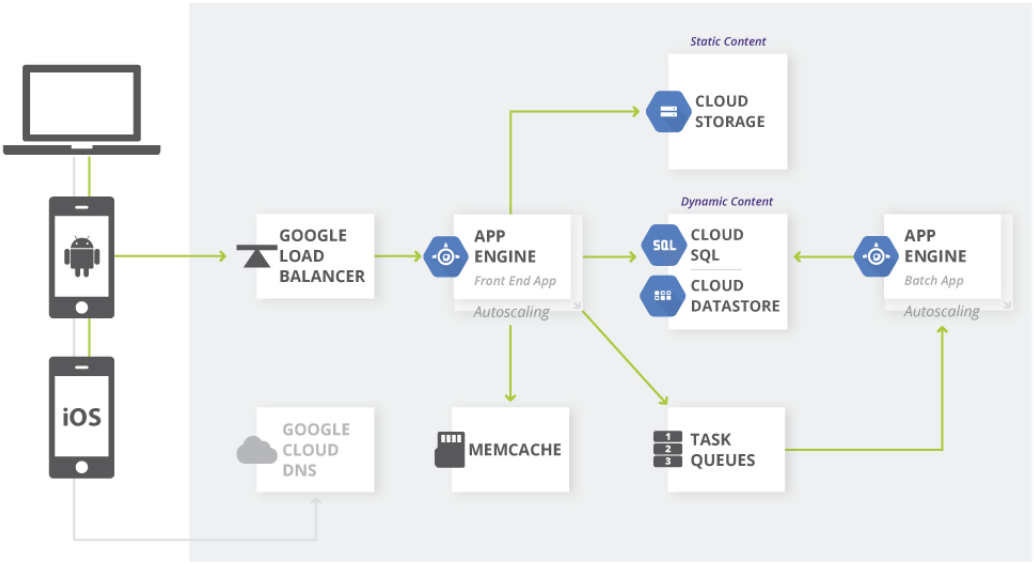
Serverová část aplikace je implementována s využitím platformy Google Cloud. Obrázek 12 zobrazuje obecné schéma architektury aplikace využívající právě tuto platformu. Datová vrstva zahrnuje Cloud Datastore a Cloud Storage, které jsou popsány v kapitole 7.2. Databáze Cloud SQL není v aplikaci využita. Zbývající komponenty uvedené ve schématu jsou využívány aplikační vrstvou. Můžeme si všimnout, že App Engine slouží také pro zpracování požadavků z fronty (*Task Queues*) na pozadí. Vlastnosti App Engine jsou uvedeny v kapitole 7.1.

## 8.2 Uživatelské rozhraní

Uživatelské rozhraní je zobrazováno webovým prohlížečem a skládá se z HTML, CSS a JavaScriptových souborů, jak jsme již zmínili v kapitole 6.4. Struktura HTML dokumentů je založena na Jinja šablonovacím systému. Webový server vyhodnocuje a vytváří kompletní HTML soubory z šablon a následně je poskytuje klientovi. Příklad takové šablony je uveden v kódu 2.

HTML šablony jsou umístěny v adresáři TEMPLATES/. Společný základ tvoří soubor BASE.HTML, který obsahuje definici rozložení prvků, import společných CSS a JavaScriptových souborů, záhlaví s navigační lištou a zápatí webu. Všechny ostatní šablony rozšiřují tento HTML dokument, a to zejména o obsah, který se zobrazuje uživateli.

Seznam lokalit je definován v LOCATION_LIST.HTML a formulář pro jejich vytváření a editaci v LOCATION_FORM.HTML. Obdobná struktura je využita pro odchyty, kde CAPTURE_LIST.HTML zajišťuje zobrazení požadovaných odchytů a ve formuláři CAPTURE_FORM.HTML je lze vytvářet a editovat. Můžeme si všimnout dvojího využití jedné šablony, která je stejná pro vytváření i editaci objektů. Aplikační část rozhoduje, jaké konkrétní hodnoty do šablony doplní a jak s nimi bude následně zacházet.




Obrázek 12: Architektura serverové části aplikace. Převzato z [28].

Zobrazování infromací týkajících se čolků je rozdělono do více šablon než v případě lokalit nebo odchytů. Soubory NEWT_UPLOAD_FORM.HTML a NEWT_CROP_FORM.HTML slouží k zobrazování uživatelského rozhraní pro nahrávání a ořez nahrané fotografie čolka. NEWT_DETAILS_FORM.HTML je šablona pro formulář, ve kterém se definují a editují podrobnosti k nově přidané fotografii. K zobrazení seznamu všech čolků je určená šablona NEWT_LIST.HTML. Detail jednoho konkrétního čolka zvoleného ze seznamu zobrazuje šablona definovaná v souboru NEWT_VIEW.HTML.

Vzhled a formátování zajišťují kaskádové styly definované v CSS souborech, které se nachází v adresáři /STATIC/CSS/. Framework Bootstrap, popsaný v kapitole 6.5, využívá také soubory z adresáře /STATIC/SCSS/. Názvy souborů, které se využívají jen pro konkrétní HTML šablonu, se shodují s jejím názvem. Příkladem může být NEWT_FORM.CSS pro NEWT_FORM.HTML. Většina kaskádových stylů je ale využita napříč celou webovou aplikací, zejména pak Bootstrap styly a jejich rozšíření pro Material Design. Zdrojové kódy třetích stran vždy obsahují odkaz na licenční ujednání v jehož souladu jsou využívány.

Obdobná situace platí pro JavaScriptové soubory, které se nachází v adresáři /STATIC/JS/. Některé části JavaScriptového kódu jsou definovány přímo v HTML šablonách z důvodu využití šablonovacího systému Jinja. Vlastnosti JavaScriptu jsou popsány v kapitole 6.4.

## 8.3 Uložení dat

Aplikace obecně pracuje se dvěma typy dat. Jedná se o fotografie v podobě binárních dat a databázové entity. Pro efektivní práci s daty nejsou fotografie ukládány do databáze, ale do Cloud Storage (viz kapitola 7.2). Společně s binárními daty fotografie ukládáme i její název, velikost a typ. Pro každou nově vloženou fotografii je vygenerován URL odkaz, který se přidá k ostatním údajů vytvářené databázové entity. V databázi je tedy uložen vždy jen odkaz na binární data fotografií. Práce s fotografiemi je definována v souboru STORAGE.PY, v němž konkrétně funkce upload_file provádí nahrávání fotografie a navrací URL k objektu v Cloud Storage, který dále využívá aplikační vrstva aplikace.

Druhým typem dat jsou databázové objekty. K manipulaci a ukládání těchto objektů využívá aplikace Cloud Datastore. V souboru MODEL_DATASTORE.PY jsou definovány funkce pro vytváření, čtení, úpravy a mazání všech typů využitých objektů. Databázový model odpovídá již popisovanému členění aplikace z uživatelského pohledu a obsahuje 4 typy objektů: *Location*, *Capture*, *Newt* a *Match*.

- *Location* slouží k uložení lokality. Obsahuje například atributy pro název lokality, zeměpisnou šířku a délku, nadmořskou výšku, počet odchytů a odkaz na obrázek lokality.
- *Capture* obsahuje potřebné atributy k uložení informací o každém odchytu. Jedná se zejména o odkaz na lokalitu, datum, jméno autora a počet odchycených čolků.
- *Newt* reprezentuje čolka. Kromě údajů vyplňovaných ve formuláři při vytváření obsahuje tento typ objektů také atributy pro odkaz na originální fotografii, fotografie vytvořené prahování a příznak toho, zda byl daný objekt chycen již vícekrát.
- *Match* je poslední využívaný typ objektu, který se vytváří při nalezení shody mezi čolky. Zahrnuje informace o míře shody, o tom, zda byla potvrzena a případně kým byla potvrzena. Samozřejmě obsahuje i odkazy na dva čolky, kterých se nalezená shoda týká. Popsané závislosti mezi objekty jsou založeny na ukládání jednoznačných identifikátorů (ID) odkazovaných objektů.

## 8.4 Aplikační vrstva

Podstatná část aplikace se nachází mezi prezentační a datovou vrstvou, kteroá byla právě popsána. Následuje přehled souborů a nejdůležitějších funkcí, které zajišťují aplikační logiku. Přiložené zdrojové kódy obsahují komentáře k jednotlivým funkcím a poskytují větší detail.

- MAIN.PY vytváří aplikaci funkcí create_app, která je definována v souboru __INIT__.PY s nastavením uloženém v CONFIG.PY. Dále se zde získává odkaz na frontu úloh, které server zpracovává na pozadí. Při spuštění aplikace na Google Cloud platformě se tyto úlohy provádí na speciální instaci App Engine, jak je vidět na obrázku 12. Lokálně se zpracování těchto úloh spouští příkazem `.\psqworker main.newts_queue`. Nakonec je zde definováno spuštění samotné aplikace na lokální adrese počítače na portu 8080, který lze samozřejmě změnit. Pro produkční verzi aplikace se využívá běhová konfigurace uvedená v APP.YAML.

- __INIT__.PY obsahuje definici již zmíněné funkce create_app, ve které je mimo jiné specifikováno, jakým způsobem se budou zpracovávat požadavky na server. Podle začátku URL za adresou serveru se požadavky začínající „/locations“ zpracovávájí v LOCATION__CRUD.PY, „/captures“ v CAPTURE__CRUD.PY a „/newts“ v NEWT__CRUD.PY. Dále funkce get_model vybírá nastavenou databázi podle konfigurace aplikace a vrací odkaz na soubor, ve kterém se nachází funkce pracující se zvoleným typem databáze. V našem případě se jedná o MODEL__DATASTORE.PY, jak již bylo uvedeno v popisu datové vrstvy aplikace. Poslední funkce v tomto souboru získává informace o přihlášeném uživateli a ukládá si je pro pozdější použití.

- CONFIG.PY definuje základní nastavení aplikace. Jedná se například o ID projektu, název databáze, povolené soubory pro nahrávání do Cloud Storage a také autentizaci pro přístup k vyjmenovaným zdrojům.

- LOCATION__CRUD.PY je první ze souborů, které určují způsob zpracování konkrétních požadavků na webový server. Operace, které souvisí s lokalitami jsou řešeny zde deklarovanými funkcemi. Pro ilustraci uveďme funkci add_location, která vytváří novou lokalitu v databázi a dále například funkci edit_location, která upravuje dříve vytvořenou lokalitu.

- CAPTURE__CRUD.PY má obdobnou strukturu jako LOCATION__CRUD.PY a zpracovává všechny požadavky týkající se odchytů. Připoměňme například, že při vytváření nového odchytu se nabízí v rozbalovací nabídce všechny lokality. Zvolené ID lokality se ukládá společně s ostatními údaji vytvářeného odchytu. Data pro formuláře založení a úpravy lokality získávají funkce uvedené právě zde.

- NEWT__CRUD.PY řeší požadavky, které souvisí s čolky. Funkce v tomto souboru často využívají funkcionalitu definovanou v TASKS.PY a DATASTORE__ENTITY__FUNCTIONS.PY. Například po uložení detailů nového čolka se zvýší počet odchycených jedinců u odpovídajícího odchytu, vyhledají se kontury na nahrané fotografii a proběhne pokus o nalezení shodné fotografie. To vše zajišťuje funkce after_newt_details_saved, uvedená v souboru details_form_post, po uložení nového objektu čolka do databáze.

- TASKS.PY obsahuje skupinu metod, které jsou využívány funkcemi z výše popsaných souborů. Většinou jde o řešení komplexnějších úloh - například nahrání fotografií, určení odhadu pohlaví čolka s využitím neuronové sítě nebo nalezení kontur na obrázku a určení podobných čolků. Také je zde iniciována fronta úloh, které jsou zpracovávány na pozadí.

- DATASTORE_ENTITY_FUNCTIONS.PY jsou funkce, které jistým způsobem souvisí s databázovými objekty. Tyto funkce reagují na změnu dat a upravují odpovídající informace v databázových objektech. K úpravám jsou využívány funkce datového modelu obsažené v MODEL_DATASTORE.PY. Příkladem může být funkce pro výpočet populace, která při přidání nového čolka nebo potvrzení shody vypočítá a aktualizuje odhadovanou populaci v databázovém objektu dané lokality.

- THRESHOLD je cloudová funkce pro prahování fotografií. Implementuje metody zpracování obrazu, které jsme představili v kapitole 4. Výsledkem je odkaz na novou fotografii, která je uložena do Cloud Storage.

- GET_SIMILARITY_SCORE je rovněž cloudová funkce, která určuje míru podobnosti jednotlivých fotografií čolků. Jedná se o klíčovou funkci aplikace. Její využívání zajišťuje funkce find_matches v souboru TASKS.PY. Při nalezení shodných fotografií je vytvořen objekt typu *Match*, který propojuje dva dané čolky spolu s informací o míře podobnosti porovnávaných fotografií.

# Závěr

Cílem práce bylo navrhnout a implementovat vhodný algoritmus, který nalezne a následně porovná charakteristické rysy čolka velkého na fotografiích pořízených během jeho odchytu. Algoritmus je implementován ve webové aplikaci, která umožňuje uživateli kategorizovat fotografie čolků podle místa a data odchytu, pohlaví nebo dle libovolně definovatelného označení. Při nahrávání nové fotografie čolka využívá aplikace metod strojového učení pro určení kvality fotografie a k návrhu rozměrů pro ořezání fotografie na tu část, v níž se nachází čolek. Neuronová síť je také využita pro určení pohlaví jedince na fotografii.

Porovnávání je založeno na segmentaci skvrn čolka, které mají unikátní tvar, velikost, počet a vzájemnou polohu. Výsledkem srovnání je tedy rozdíl v podobnosti porovnávaných fotografií. Aplikace provádí automatické srovnání fotografií čolků stejného pohlaví pořízených ve stejné lokalitě v různé dny. Stejným způsobem se provádělo manuální porovnávání fotografií v praxi doposud. Pomocí metody zpětného odchytu je nakonec vypočítána odhadovaná velikost populace v dané lokalitě.

K vytvoření webové aplikace byl zvolen programovací jazyk Python, webový framework Flask, grafická knihovna OpenCV a standardní webové technologie zahrnující HTML, CSS a JavaScript. Aplikace také využívá některé nástroje z platformy Google Cloud. Jedná se například o databázi, cloudové funkce nebo App Engine umožňující hostování aplikace bez nutnosti správy serverů a infrastruktury, na které je aplikace provozována.

Aplikace byla otestována na fotografiích z pěti různých dní odchytu. Někteří jedinci se vyskytovali současně na více fotografiích a aplikace vyhodnotila tyto fotografie jako nejpodobnější v 86 %. U zbývajících fotografií byla kontrolována správnost nabízených shod. V 92 % se názor tří nezávislých posuzovatelů shodoval s výsledky algoritmu.

Architektura aplikace umožňuje její snadné rozšíření pro podporu dalších druhů živočichů, u nichž se provádí výzkum a odhad velikosti populace pomocí metody zpětného odchytu. Dalším možným vylepšením by mohlo být zavedení uživatelských rolí a oprávnění, které by se využívalo například pro omezení práv pro studenty nebo schvalování nahrávaných fotografií zvolenými uživateli. V případě využívání aplikace napříč organizacemi by bylo také vhodné přidat možnost zobrazovat pouze data určité organizace.

# Conclusions

The aim of the thesis was to design and implement a suitable algorithm for finding and comparing the characteristic features of the Great Crested Newt in photographs taken during its capture. The algorithm is implemented in a web application that allows a user to categorize newts' photos by location, capture date, sex or by user-defined tag. During an upload of a new photo, the application uses machine learning methods to determine the quality of the photo and also to determine the position of the animal and suggest the way to crop the photo. The neural network is also used to determine the sex of an individual in a photograph.

The comparison is based on the segmentation of spots on the body of the newt. These spots have a unique shape, size, number, and relative position. The result of the comparison is the difference in the similarity of the photos compared. The application performs an automatic comparison of same-sex newt photos taken at the same location on different days. Comparisons have been made manually in a similar manner until now. Finally, the estimated population size is calculated by the capture-recapture method.

The Python programming language, the Flask web framework, the OpenCV graphics library, and standard web technologies including HTML, CSS, and JavaScript were used to create the web application. The app also utilizes tools from Google Cloud platform. These tools include databases, Cloud Functions, and Google App Engine.

Test of the application was performed in photographs taken during 5 days of capture. In 86 %, different photographs with the same individuals were considered to be the most similar. The remaining photos were checked for the correctness of the offered matches. In 92 %, the opinion of three independent reviewers matched with the results of the algorithm.

The application architecture is ready for an extension to support other animal species that are researched using the capture-recapture method for estimation of population size. Another possible improvement could be the introduction of user roles and permissions that would be used, for example, to restrict student rights or to approve uploaded photos by selected users. When using an application across organizations, it would also be a good idea to add the ability to display only the data of your own organization.

# A Obsah přiloženého CD

Přiložené CD má následující adresářovou strukturu a obsah:

**`bin/`**
: Zdrojové soubory potřebné pro lokální spuštění webové aplikace.

**`doc/`**
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh a souborů potřebných pro vygenerování PDF dokumentu textu.

**`src/`**
: Kompletní zdrojové soubory aplikace, včetně kopií zdrojových kódů cloudových funkcí a testovacích fotografií čolka.

**`readme.txt`**
: Instrukce pro lokální spuštění aplikace a odkaz na veřejně dostupnou verzi aplikace.

# Literatura


[1] AmphIdent: Pattern Matching and Automatic Photo-Identification of Individual Amphibians [online]. 2018 [cit. 2019-04-04]. Dostupné z: http://www.amphident.de/en/

[2] Bootstrap [online]. c2010-2019 [cit. 2019-04-04]. Dostupné z: https://getbootstrap.com/

[3] CSS [online]. c1999-2019 [cit. 2019-03-27]. Dostupné z: https://www.w3schools.com/

[4] CVRČEK, Václav. PRECISION & RECALL. In: Petr Karlík, Marek Nekula, Jana Pleskalová (eds.), CzechEncy - Nový encyklopedický slovník češtiny [online]. 2017 [cit. 2019-04-09]. Dostupné z: https://www.czechency.org/slovnik/PRECISION & RECALL

[5] Flask [online]. c2010-2019 [cit. 2019-03-25]. Dostupné z: http://flask.pocoo.org/

[6] FLUSSER, Jan, SUK, Tomás a ZITOV, Barbara. Moments and moment invariants in pattern recognition. Hoboken, N.J.: J. Wiley, 2009. ISBN 978-0-470-69987-4.

[7] GONZALEZ, Rafael C. a WOODS, Richard E. Digital image processing. 3rd ed. Upper Saddle River, N.J.: Prentice Hall, c2008. ISBN 978-0131687288.

[8] Google Cloud [online]. c2019 [cit. 2019-04-07]. Dostupné z: https://cloud.google.com/

[9] Google Cloud Functions [online]. c2019 [cit. 2019-04-07]. Dostupné z: https://cloud.google.com/functions/

[10] GURNEY, Kevin. An introduction to neural networks. London: UCL Press, 1997. ISBN 18-572-8673-1.

[11] HOLČÍK, Jiří, KOMENDA, Martin (eds.) a kol. Matematická biologie: e-learningová učebnice [online]. 1. vydání. Brno: Masarykova univerzita, 2015. ISBN 978-80-210-8095-9.

[12] CHAPMAN, Dougulas G. Some properties of the hypergeometric distribution with applications to zoological sample censuses. Berkeley: University of California Press, 1951.

[13] JACKSON, Philip C., ABRAMSON, J. H., ed. Introduction to artificial intelligence. 2nd, enl. ed. New York: Dover, 1985. ISBN 04-862-4864-X.

[14] JavaScript [online]. c2016-2019 [cit. 2019-03-27]. Dostupné z: https://www.javascript.com/

[15] Jinja2 [online]. c2014 [cit. 2019-03-25]. Dostupné z: http://jinja.pocoo.org/

[16] JOLLY, G M. Explicit estimates from capture-recapture data with both death and immigration-stochastic model. Biometrika 52:225-247, 1965.

[17] KUZMIN, Sergius. Amphibians of the Former Soviet Union. Sofia: Pensoft Publ., 1999. ISBN 95-464-2045-X.

[18] KREBS, Charles J. Ecological Methodology. New York: Harper Collins Publishers, 1989. ISBN 00-604-3784-7.

[19] KRÖSE, Ben a VAN DER SMAGT, Patrick. An introduction to neural networks. Eight edition. Amsterdam: The University of Amsterdam, 1996.

[20] LAST, John M. a ABRAMSON, J. H. A dictionary of epidemiology. 3rd ed. New York: Oxford University Press, 1995.

[21] Material Design [online]. c2019 [cit. 2019-04-04]. Dostupné z: https://material.io/

[22] Material Design for Bootstrap [online]. c2019 [cit. 2019-04-04]. Dostupné z: https://mdbootstrap.com/

[23] OpenCV [online]. c2019 [cit. 2019-03-20]. Dostupné z: https://opencv.org/

[24] OTIS, David L., BURNHAM, Kenneth P., WHITE, Gary C. a ANDERSON, David R. Statistical Inference from Capture Data on Closed Animal Populations. Wildlife Monographs. Kansas: Allen Press, 1978.

[25] Python [online]. c2001-2019 [cit. 2019-03-20]. Dostupné z :https://www.python.org/

[26] ResearchGate: Schematic drawing of multilayer perceptron neural networks. In: ResearchGate [online]. Berlín, c2019 [cit. 2019-03-28]. Dostupné z: https://www.researchgate.net/

[27] ROČEK, Zbyněk. Triturus alpestris. In: BARUŠ, V., OLIVA, O. (eds) Fauna Československa - Amphibia. Academia Praha, 1992.

[28] Serving websites. Google Cloud [online]. c2019 [cit. 2019-04-20]. Dostupné z: https://cloud.google.com/solutions/web-serving-overview#app-engine

[29] SHALEV-SHWARTZ, Shai a BEN-DAVID, Shai. Understanding machine learning: from theory to algorithms. 2nd, enl. ed. New York, NY, USA: Cambridge University Press, 2014. ISBN 978-1-107-05713-5.

[30] SMITH, Christopher U. M. Elements of molecular neurobiology. 3rd ed. Hoboken, N.J.: J. Wiley, c2002. ISBN 0-470-84353-5.

[31] Vývojářská dokumentace OpenCV. Color conversion [online]. c2018 [cit. 2019-03-12]. Dostupné z: https://docs.opencv.org/3.4

[32] Vývojářská dokumentace OpenCV. Image Thresholding [online]. c2018 [cit. 2019-03-10]. Dostupné z: https://docs.opencv.org/3.4

[33] Vývojářská dokumentace OpenCV. Structural Analysis and Shape Descriptors [online]. c2018 [cit. 2019-03-27]. Dostupné z: https://docs.opencv.org/3.4