



## ZADÁNÍ BAKALÁ SKÉ PRÁCE

<b>Název:</b>	Detekce objektů v ptáčích hnízdech pomocí neuronových sítí
<b>Student:</b>	Jan Havlík
<b>Vedoucí:</b>	Ing. Josef Pavlásek, Ph.D.
<b>Studijní program:</b>	Informatika
<b>Studijní obor:</b>	Softwarové inženýrství
<b>Katedra:</b>	Katedra softwarového inženýrství
<b>Platnost zadání:</b>	Do konce letního semestru 2017/18

### Pokyny pro vypracování

Navhněte a implementujte knihovnu umožňující zpracovávat data z kamery umístěné v hnízdě (ptačí budce) s cílem rozpoznat počet vajec v hnízdě. Pro rozpoznání použijte existující algoritmy umělé inteligence využívající data získaná ze serveru PtaciOnline.cz a další sv. nástroje projektu BirdObserver (athena.pef.czu.cz).

Postupujte v těchto krocích:

1. Provejte detailní specifikaci požadavku.
2. Seznámte se s projektem BirdObserver a strukturou dat na serveru PtaciOnline.cz.
3. Provejte analýzu a návrh knihovny.
4. Návrh implementujte, zdokumentujte a vhodným způsobem otestujte.
5. Knihovnu navrhněte a implementujte v jazyce Java tak, aby bylo možné ji formou závislostí (dependency) integrovat s ostatními nástroji projektu BirdObserver.

### Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.  
vedoucí katedry

prof. Ing. Pavel Tvrďák, CSc.  
d. kan

V Praze dne 12. ledna 2017





**FAKULTA  
INFORMAČNÍCH  
TECHNOLOGIÍ  
ČVUT V PRAZE**

Bakalářská práce

## **Detekce objektů v ptačích hnízdech pomocí neuronových sítí**

*Jan Havlůj*

Katedra softwarového inženýrství  
Vedoucí práce: Ing. Josef Pavláček, Ph.D.

2. ledna 2018



---

## **Poděkování**

Chtěl bych poděkovat vedoucímu mé bakalářské práce, panu Ing. Josefovi Pavláčkovi Ph.D. za jeho čas, ochotu a pomoc při vedení této práce. Děkuji svým rodičům a přítelkyni, kteří mi byli po celou dobu studia velkou oporou. V neposlední řadě bych chtěl poděkovat svým spolužákům, kteří mě při studiu nikdy neváhali podpořit.



---

## **Prohlášení**

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 2. ledna 2018

.....

České vysoké učení technické v Praze  
Fakulta informačních technologií  
© 2018 Jan Havlůj. Všechna práva vyhrazena.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.*

#### **0.0.1 Odkaz na tuto práci**

Havlůj, Jan. *Detekce objektů v ptačích hnizdech pomocí neuronových sítí*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018. Dostupný také z WWW: <[https://github.com/havluj/bachelor\\_thesis](https://github.com/havluj/bachelor_thesis)>.

---

# Abstrakt

Tato práce se zabývá návrhem a tvorbou softwarové knihovny pro detekci objektů v obrazu použitím neuronových sítí.

V první části jsme seznámeni s cílem práce a detailní specifikací požadavků. Druhá část se věnuje teorii počítačového vidění, strojového učení a neuronových sítí. Následně je na základě získaných teoretických znalostí a specifikace požadavků vypracována analýza možného řešení. Dle analýzy jsou navržena a implementována dvě řešení. První implementace se zaměřuje na detekci objektů, druhá na rozpoznávání a klasifikaci obrazu. V poslední části je vybráno efektivnější řešení, které je řádně ověřeno a otestováno.

Výsledkem je softwarová knihovna, která umožňuje automaticky rozpoznávat počet vajec v daném videu. Celý program je implementován v jazyce Java a je možné ho integrovat do projektu BirdObserver.

**Klíčová slova** neuronové sítě, počítačové vidění, detekce objektů, strojové učení, učení s učitelem, rozpoznávání obrazu, detekce vajec, ptačí hnízda, Java

---

# Abstract

This thesis focuses on designing and creating a software library that is able to detect objects in an image by using neural networks.

In the first part of the thesis, goals are set technical parameters are specified. The second part discusses the theory behind computer vision, machine learning, and neural networks. Using the technical specification and acquired theoretical knowledge, a detailed analysis of a possible solution is presented. There are two implementations of the analysis. The first one is using object detection to meet it's goal, the second one image recognition. In the last part of the thesis, a more effective implementation is chosen, which is then properly tested and verified.

The result of this thesis is a software library that is able to automatically detect the number of eggs in a given video sequence. The entire solution is written in Java and is easily intergratable with the BirdObserver project.

**Keywords** neural networks, computer vision, object detection, machine learning, supervised learning, image recognition, egg detection, bird nests, Java

---

# Obsah

<b>Úvod</b>	<b>15</b>
Struktura práce . . . . .	15
<b>1 Cíl práce</b>	<b>17</b>
1.1 Nefunkční požadavky . . . . .	17
1.2 Funkční požadavky . . . . .	17
<b>2 Rešerše</b>	<b>19</b>
2.1 Počítačové vidění . . . . .	19
2.2 Segmentace obrazu . . . . .	19
2.3 Detekce objektů . . . . .	19
2.4 Klasifikace obrazu . . . . .	19
2.5 Neuronové sítě . . . . .	19
2.6 Tensorflow . . . . .	19
<b>3 Analýza a návrh</b>	<b>21</b>
3.1 Ptáci Online . . . . .	21
3.2 Představení vstupních dat . . . . .	21
3.3 Současný způsob zpracování dat . . . . .	22
3.4 Nefunkční požadavky . . . . .	23
3.5 Funkční požadavky . . . . .	23
3.6 Návrh řešení . . . . .	25
3.7 Volba technologií . . . . .	26
3.8 Shrnutí kapitoly . . . . .	27
<b>4 Nástroje</b>	<b>29</b>
4.1 Hromadné stažení dat . . . . .	29
4.2 Příprava trénovacích a testovacích dat . . . . .	30
<b>5 Implementace detekování objektů</b>	<b>35</b>
5.1 Příprava dat . . . . .	35
5.2 Trénování neuronové sítě . . . . .	35
5.3 Ověření funkčnosti . . . . .	35

<b>6</b>	<b>Implementace rozpoznávání obrazu</b>	<b>37</b>
6.1	Příprava dat . . . . .	37
6.2	Trénování neuronové sítě . . . . .	38
6.3	Ověření funkčnosti . . . . .	39
6.4	Identifikované problémy . . . . .	39
<b>7</b>	<b>Volba řešení</b>	<b>41</b>
7.1	Srovnání implementací . . . . .	41
7.2	Exportování grafu . . . . .	41
7.3	Implemetace Java knihovny . . . . .	41
<b>8</b>	<b>Ověření implementace</b>	<b>43</b>
	Závěr	45
	Literatura	47
<b>A</b>	<b>Seznam použitých zkratek</b>	<b>49</b>
<b>B</b>	<b>Dokumentace API knihovny</b>	<b>51</b>
B.1	Package org.cvut.havluja1.eggdetector . . . . .	51
<b>C</b>	<b>Tagger</b>	<b>57</b>
C.1	Dokumentace (v AJ) . . . . .	57
C.2	Zdrojový kód . . . . .	57
C.3	Ukázkový výstup . . . . .	57
<b>D</b>	<b>Folder Trimmer</b>	<b>59</b>
D.1	Dokumentace (v AJ) . . . . .	59
D.2	Zdrojový kód . . . . .	59
<b>E</b>	<b>Použité programy</b>	<b>63</b>
<b>F</b>	<b>Obsah přiloženého CD</b>	<b>65</b>

---

## **Seznam obrázků**

3.1	Ukázka neupraveného snímku z ptačí budky.	22
3.2	Vejce nemusí být vždy viditelná.	22
3.3	Program určený ke klasifikaci obrazu.	25
4.1	Tagger – hromadné určování počtu vajec.	31
4.2	Uživatelské rozhraní nástroje LabelImg.	33



---

# **Seznam algoritmů**

3.1	Prvotní návrh uživatelského rozhraní knihovny.	24
4.1	Hromadné stažení dat ze serveru athena.pef.czu.cz.	29
4.2	Výběr vhodných složek pro trénování nástrojem Tagger.	32
	media/labelimg.xml	33
	media/imgdata.xml	57
D.1	Zdrojový kód nástroje FolderTrimmer	60



---

# Úvod

Projekt Ptáci Online byl spuštěn Fakultou životního prostředí České zemědělské univerzity v Praze roku 2014 [1]. Hlavním cílem projektu je poskytnout vědecká data, ve formě videa z ptačích budek, široké veřejnosti [1]. Projekt se těší poměrně velké popularitě [2] a aktuálně spolupracuje s více než dvěma desítkami spolupracovníků. Mezi ně patří lidé z akademické sféry, soukromého sektoru, ale i z Ministerstva životního prostředí České republiky [1]. Vzhledem k množství pořízených záznamů, by bylo žádoucí informace extrahovat automaticky pomocí algoritmů umělé inteligence. Předmětem této práce je vytvořit algoritmus, který je schopný určit počet vajec v hnizdě v daném videozáznamu.

Existuje hned několik způsobů, jak naučit počítač „vidět“. Téměř vždy se musíme zaměřit na předzpracování obrazu, jeho vlastnosti a jeho segmentaci. Jako řešení pak můžeme zvolit různou sadu deterministických algoritmů, například pro detekci hran nebo na samotné klasifikování segmentovaného obrazu. Všechny tyto algoritmy mají však jednu společnou nevýhodu. Formálně popsat tvar nějakého objektu a vytvořit sadu pravidel, podle kterých poznáme, jestli se jedná o hledaný objekt, je velmi těžké. Světelné podmínky nemusí být ideální, objekty se mohou překrývat, mohou být různě barevné, vzdálené nebo otočené. Všechny tyto problémy znemožňují vytvoření perfektního algoritmu manuálně. Lepším řešením je manuálně vytvořit jeden algoritmus, který dokáže „vytrénovat“ druhý obecný algoritmus k vyřešení konkrétního problému. Za tímto účelem vznikly algoritmy inspirované přírodou, například umělé neuronové sítě inspirované lidským učením a mozkem nebo evoluční algoritmy inspirované evoluční teorií. V této práci se budeme soustředit na použití neuronových sítí pro vyřešení problémů počítačového vidění.

## Struktura práce

Práce je rozdělena do 7 částí. První část obsahuje stanovení cílů a specifikaci funkčních i nefunkčních požadavků. Druhá část je teoretická. Diskutuje problematiku počítačového vidění, strojového učení a neuronových sítí. Třetí, analytická část, se zaměřuje na výběr vhodných technologií a postupů pro tvorbu implementace. Čtvrtou i pátou, praktickou část, tvoří dva různé způsoby implementace, jejich porovnání, otestování a validace. Na závěr vybereme efektivnější implementaci, dle které je vytvořena softwarová knihovna v jazyce Java.



# Cíl práce

Cílem teoretické části práce je se seznámit s technologiemi a principy, které jsou dnes standardně používány k řešení problémů spjatých s počítačovým viděním a strojovým učením. Zaměříme se především na umělé neuronové sítě, jejich historii, současný vývoj a hlavně na principy jejich fungování. Na základě získaných teoretických znalostí je vypracována analýza řešení, která může být považována za výstup teoretické části. Cílem analýzy je jednak selekce vhodných principů a technologií pro implementaci řešení, ale i zpracování technických požadavků zároveň s popsáním současného stavu.

Cílem praktické části je implementovat řešení, které bude dostatečně rychlé a spolehlivé. Výstupem bude softwarová knihovna, která bude umožňovat rozpoznávat počet vajec v hnázde na jednotlivých snímcích i na sekvenčních videa. Součástí knihovny bude neuronová síť, která bude řešit samotnou detekci. Tento přístup umožní recyklaci výsledného programu s minimální modifikací pro řešení podobných problémů, jako například určení druhu ptáka, počtu mláďat nebo predaci hnázda.

Následující seznam funkčních a nefunkčních požadavků slouží k přesné definici požadavků na výsledný systém. Tento seznam byl vytvořen ve spolupráci s vedoucím práce.

## 1.1 Nefunkční požadavky

**N1** Knihovna musí pracovat v reálném čase. Po zadání vstupních dat je výsledek očekáván maximálně v jednotkách sekund.

## 1.2 Funkční požadavky

**F1** Systém bude distribuovaný formou Java knihovny v archivu JAR<sup>1</sup>.

**F2** Knihovna bude distribuována ve dvou verzích:

- JAR archiv obsahující jak přeložený zdrojový kód knihovny, tak i všechny závislosti<sup>2</sup>, které jsou knihovnou vyžadovány.

<sup>1</sup>Zkratka pro Java ARchive. „JAR je kompresní souborový formát, používaný platformou Java, založený na ZIP kompresi.“[3]

<sup>2</sup>Zdrojový kód může používat funkce poskytované jinými softwarovými knihovnami.

## 1. CÍL PRÁCE

---

- JAR archiv obsahující pouze přeložený zdrojový kód knihovny bez externích závislostí. Do archivu bude přibalen konfigurační soubor pro systém Maven[4] s definicí závislostí.

**F3** Knihovna musí umět pracovat se strukturou dat na serveru <http://athena.pef.czu.cz/ptacionline/>.

**F4** Knihovna musí umět určit počet vajec v hnízdě pro každy jednotlivý snímek.

**F5** Knihovna musí umět určit počet vajec v hnízdě pro složku jako celek. Složka se skládá ze sekvence obázků, které reprezentují jednu videosekvenci.

**F6** Snímky mohou být ve formátu JPEG<sup>3</sup> a PNG<sup>4</sup>.

---

<sup>3</sup>Joint Photographic Experts Group.

<sup>4</sup>Portable Network Graphics.

## Rešerše

**2.1 Počítačové vidění**

**2.2 Segmentace obrazu**

**2.3 Detekce objektů**

**2.4 Klasifikace obrazu**

jak funguje

**2.5 Neuronové sítě**

**2.5.1 CNN**

**2.5.2 RCNN**

**2.6 Tensorflow**

Text



## Analýza a návrh

V této kapitole se zaměříme na strukturu projektu Ptáci Online, problémy spojené se současným způsobem sbírání dat, rozbor funkčních a nefunkčních požadavků, návrh řešení a výběr technologií. Na závěr budeme diskutovat dva různé způsoby řešení – jejich výhody a nevýhody.

### 3.1 Ptáci Online

„Cílem projektu je popularizovat ochranu ptáků v blízkosti lidských sídel, jejich hnízdění, včetně jeho monitoringu, s využitím speciálního technického zařízení tzv. „chytré ptačí budky“.“ [5]

Tyto „chytré ptačí budky“ slouží k monitorování hnízdícího ptactva. Každá budka obsahuje jednu nebo dvě kamery s nočním přísvitem. Ve vletové otvoru budky je umístěn pohybový senzor, který spustí natáčení kamér při detekci pohybu. Dále je do budky vestavěn venkovní a vnitřní teplotní senzor, mikrofon a senzor venkovního osvětlení, který reguluje funkci přísvitu kamér. Přenos nasbíraných dat z budky probíhá přes ethernetový PoE<sup>5</sup> kabel. Tento kabel zajišťuje i napájení veškeré elektroniky uvnitř budky. [1] Jak vypadá záznam z budky je vidět na obr. 3.1.

### 3.2 Představení vstupních data

Ptačí budka vyprodukuje sekvenci videa pokaždé, když je v ní detekován pohyb. Kvůli energetické úspornosti kamer a množství přenášených dat mají tyto videa nižší snímkovou frekvenci<sup>6</sup>. Z takového videozáznamu lze extrahovat množinu jednotlivých snímků, které jako celek tvoří dané video. Snímky jsou ukládány do formátu PNG a zařazeny do složek, přičemž **jedna složka reprezentuje jeden videozáznam**.

Výsledná softwarová knihovna bude umět pracovat s jednotlivými složkami. Načte všechny snímky z dané složky a následně je zpracuje. Uživatel bude schopen získat informace o složce jako celku i jednotlivých snímcích.

---

<sup>5</sup>Power over Ethernet.

<sup>6</sup>Počet snímků za sekundu.

### 3. ANALÝZA A NÁVRH

---



Obrázek 3.1: Ukázka neupraveného snímku z ptačí budky.



(a) Vejce jsou zřetelně viditelná v čase 0:01.



(b) Vejce nejsou viditelná ve zbytku videa, jako je tomu např. v čase 0:14.

Obrázek 3.2: Vejce nemusí být vždy viditelná.

### 3.3 Současný způsob zpracování dat

Akademickí pracovníci potřebují nashromáždit velké množství dat, ale taková činnost je velmi časově náročná. Proto jsou na takovou práci najímáni brigádníci. Brigádníky je nutno nejprve zaškolit, aby věděli, co mají ve videích hledat. Poté sledují jedno video za druhým a získané informace zapisují do tabulek v Excelu. Tento způsob práce je drahý a časově náročný.

Jedna z informací, která nás může zajímat, je počet vajec v hnizdě. Brigádník musí video otevřít, shlédnout a najít část, kde jsou vejce zřetelně viditelná (viz obr. 3.2a). Poté vejce spočítá a výsledek zapíše do tabulky. Jelikož je nutné získat co nejvíce dat, brigádníci tento proces vykonávají co nejrychleji. To vede k chybám, např. chybně spočítaný počet vajec nebo zapsání výsledku na špatný řádek tabulky.

Je snahou vytvořit systémy, které by tyto úkoly mohly provádět automaticky. Příkladem takového systému je například práce od Pavla Šumy, který se snaží automaticky zjistit počet mláďat v hnizdě [6]. Dalším příkladem je i tato práce.

## 3.4 Nefunkční požadavky

### 3.4.1 N1

*Knihovna musí pracovat v reálném čase. Po zadání vstupních dat je výsledek očekáván maximálně v jednotkách sekund.*

Z vlastnosti neuronových sítí vyplývá, že tento požadavek nebude problém splnit. Samotný průchod snímku grafem, resp. průchod snímku neuronovou sítí není příliš časově náročný. Nutnou podmínkou však je dostupný soubor obsahující popis struktury neuronové sítě. Trénování, nebo-li hledání optimální struktury neuronové sítě je velmi výpočetně náročná činnost, která musí být dokončena **před** použitím knihovny.

## 3.5 Funkční požadavky

### 3.5.1 F1

*Systém bude distribuovaný formou Java knihovny v archivu JAR.*

Neuronové sítě můžeme naprogramovat v libovolném programovacím jazyce. Tato podmínka může být tedy bez problému splněna. Výsledná softwarová knihovna pro detekci počtu vajec v hnizdě bude implementována v jazyce Java.

### 3.5.2 F2

*Knihovna bude distribuována ve dvou verzích:*

- *JAR archiv obsahující jak přeložený zdrojový kód knihovny, tak i všechny závislosti, které jsou knihovnou vyžadovány.*
- *JAR archiv obsahující pouze přeložený zdrojový kód knihovny bez externích závislostí. Do archivu bude přibalen konfigurační soubor pro systém Maven[4] s definicí závislostí.*

Distribuci přeloženého zdrojového kódu vyřešíme pomocí nástroje pro správu, řízení a automatizaci buildů aplikací. Maven je pro nás nejhodnější volba, vzhledem k druhé části podmínky – Do archivu bude přibalen konfigurační soubor pro systém Maven s definicí závislostí.

### 3.5.3 F3

*Knihovna musí umět pracovat se strukturou dat na serveru <http://athena.pef.czu.cz/ptacionline/>.*

Knihovna bude schopna pracovat se strukturou dat popsanou v kapitole 3.2. Uživateli bude schopen interagovat s knihovnou, která bude reflektovat strukturu dat projektu:

### 3. ANALÝZA A NÁVRH

---

Algoritmus 3.1: Prvotní návrh uživatelského rozhraní knihovny.

```
1 // Inicializujeme EggDetector. Knihovna se připraví pro
2 // zpracování sekvencí.
3 EggDetector eggDetector = new EggDetector();
4
5 // EggDetectoru předáme absolutní cestu k složce, kterou
6 // chceme vyhodnotit. EggDetector nám vrátí třídu
7 // SequenceClassifier, která obsahuje veškeré informace
8 // k dané složce.
9 SequenceClassifier sequenceClassifier = eggDetector.evaluate(new File("image_dir"));
10
11 // Zjistíme finální počet vajec v hnízdě pro danou složku.
12 System.out.println("final count: " + sequenceClassifier.getFinalCount());
13
14 // Můžeme zjistit počet vajec v jednotlivých snímcích.
15 System.out.println("individual scores: " + sequenceClassifier.getIndividualCounts());
16
17 // Ukončíme EggDetector - uvolníme informace o neuronové
18 // sítě z paměti. Instance EggDetectoru se stane nepoužitelná.
19 eggDetector.closeSession();
```

---

#### 3.5.4 F4

*Knihovna musí umět určit počet vajec v hnízdě pro každý jednotlivý snímek.*

Výsledná knihovna bude uživateli umět poskytnout informace o jednotlivých snímcích – viz algoritmus 3.1.

#### 3.5.5 F5

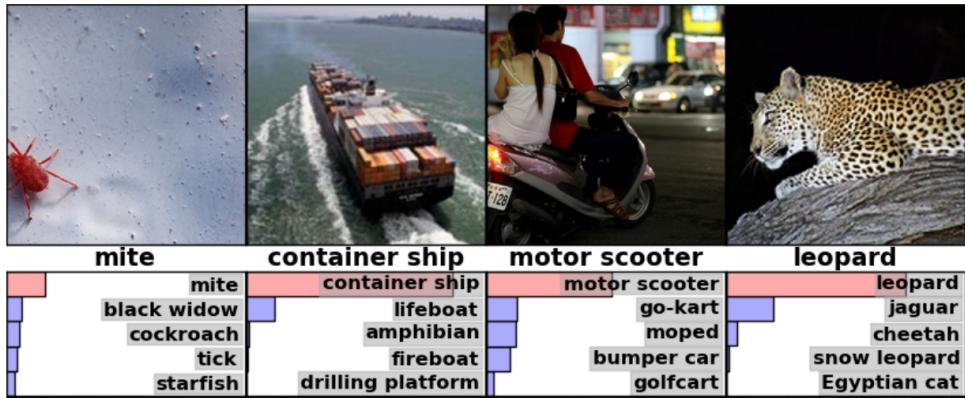
*Knihovna musí umět určit počet vajec v hnízdě pro složku jako celek. Složka se skládá ze sekvence obázků, které reprezentují jednu videosekvenci.*

Výsledná knihovna bude uživateli umět poskytnout informace o složce jako celku – viz algoritmus 3.1.

#### 3.5.6 F6

*Snímky mohou být ve formátu JPEG a PNG.*

Pro zpracování snímků použijeme standartně dostupné třídy pro práci s grafikou v programovacím jazyce Java. Konkrétně `BufferedImage` a `Graphics2D` nám umožní zpracovat oba dva formáty – JPEG i PNG.



Obrázek 3.3: Program určený ke klasifikaci obrazu.

Zdroj: dokumentace softwarové knihovny TensorFlow [7].

### 3.6 Návrh řešení

Jádro implementace bude tvořit neuronová síť, která bude vytvořena metodou „učení s učitelem“<sup>7</sup>. Tvorba implementace se bude skládat ze čtyř částí:

- Příprava trénovacích a testovacích dat.
- Trénování neuronové sítě.
- „Konzumace“ vytrénované neuronové sítě knihovnou, která je výsledkem této práce.
- Ověření funkčnosti.

Existuje několik způsobů, jak neuronovou síť vytrénovat. Prvním možným řešením je neuronová síť určená ke klasifikaci celého obrazu. Typickým vstupem je snímek, ve kterém je objekt, který chceme rozpoznat, nejvýraznější. Pokročilejší typ této sítě je schopný popsat komplexnější scény, jako například „Dva lidé na pláži.“. Ukázka takového programu je vidět na obrázku 3.3. V našem případě bychom klasifikovali počet vajec v hnizdě, kde by jednotlivé výsledky reprezentovali počet vajec na daném snímku. Tzn. výstupem takové neuronové sítě by byla jedna z předem daných kategorií, o které si s největší pravděpodobností myslíme, že popisuje, co je na daném snímku. Každá kategorie by reprezentovala jiný počet vajec v hnizdě, například kategorie 0, kategorie 1, atd.

Druhým možným řešením je komplexnější neuronová síť určená k detekci objektů. Obraz je nejprve segmentován na části, o kterých si síť myslí, že by mohly obsahovat nějaké objekty. Následně jsou tyto části klasifikovány a je rozhodnuto, zda-li se jedná o objekt a s jakou pravděpodobností. Výsledkem je potom množina detekcí, která je tvořena umístěním objektu, typem objektu a pravděpodobností mírou, která reprezentuje jak moc je si síť jistá, že se opravdu jedná o daný objekt. Typickým vstupem je libovolný snímek, jako je tomu vidět na obrázku ???. V našem případě bychom hledali detekci vajec s relativně vysokou pravděpodobností. Počet detekcí by reprezentoval počet vajec v hnizdě.

Ať už zvolíme jakýkoliv způsob implementace, pro zpracování obrazu předáme neuronové síti pole hodnot o velikosti 300x300x3 (snímek bude zmenšen na velikost 300x300 bodů a zůstane

<sup>7</sup>Supervised learning.

### 3. ANALÝZA A NÁVRH

---

barevný – každý bod obsahuje 3 barevné složky<sup>8</sup>), které reprezentuje náš snímek. Knihovna nám poté vrátí požadovaný výsledek<sup>9</sup>.

## 3.7 Volba technologií

V této kapitole vybereme konkrétní platformy, programovací jazyky a nástroje potřebné k implementaci praktické části této práce.

### 3.7.1 Platforma

Vzhledem k funkčním požadavkům je jednoznačnou volbou programovací jazyk Java. Volíme verzi Java 8 SE, která poskytuje všechny nástroje, které k doručení výsledku potřebujeme.

### 3.7.2 Neuronové sítě

Pro značné usnadnění tvorby neuronových sítí použijeme softwarovou knihovnu. Mezi tři nejznámější patří TensorFlow, OpenNN a FANN<sup>10</sup>. Z těchto tří knihoven pouze TensorFlow poskytuje Java API a už jen z tohoto důvodu je náš nejlepší kandidát.

TensorFlow má skvělou dokumentaci [8] s velkým množstvím kompletních návodů [9]. Poskytuje API k detekci objektů, kde je možné využít již předtrénované modely [10]. Pomocné skripty určené k trénování neuronové sítě, uložení její struktury a ověření funkčnosti nainplementujeme v jazyce Python 3, jelikož je primárním jazykem pro práci s knihovnou TensorFlow.

### 3.7.3 Práce s daty

K vytrénování naší neuronové sítě je potřeba velké množství trénovacích dat. Trénovacími daty jsou myšleny snímky, ke kterým dodáme informace, jako například počet a umístění jednotlivých vajíček. Abychom si získávání a označování dat usnadnili, je potřeba několik nástrojů, které jsou detailně popsány v kapitole 4.

- Pro hromadné stažení dat použijeme nástroje `bash` a `wget`. Hromadně stažená data budou obsahovat i obsah, který pro nás není užitečný. Proto použijeme nástroj, který všechna neužitečná data smaže. Více v kapitole 4.1.
- Trénovací a testovací data vytvoříme pomocí nástrojů `LabelImg` a `Tagger`. Detailní informace obsahuje kapitola 4.2.

Abychom byli schopni nově vytvořená trénovací data použít k trénování neuronové sítě, musíme je převést do standardního formátu. V našem případě musíme z binárních dat snímků a textových XML souborů vytvořit tzv. `TFRecord` [11]. Každý způsob implementace vyžaduje mírně odlišný formát trénovacích dat. V kapitolách 5 a 6, které se popisují konkrétní implementace, diskutujeme i přípravu trénovacích dat.

---

<sup>8</sup>RGB - red, green, blue. Každý bod obrázku obsahuje informaci o koncentraci červené, zelené a modré.

<sup>9</sup>V případě rozpoznávání obrazu, knihovna vrátí seznam pravděpodobností pro všechny známe typy. V případě detekce objektů, knihovna vrátí seznam, pozici a typ objektů.

<sup>10</sup>Fast Artificial Neural Network.

## 3.8 Shrnutí kapitoly

V této kapitole jsme prozkoumali strukturu projektu Ptáci Online, problémy současného způsobu sbírání dat. Adresovali jsme všechny funkční i nefunkční požadavky s ohledem na strukturu projektu a dat. Navrhli jsme 2 možná konkurenční řešení, u kterých není předem jasné, jaké z nich je efektivnější. Nezbývá nám tedy nic jiného, než implementovat obě dvě řešení a jejich výsledky porovnat. Zaměřili jsme se také na výběr vhodných principů a nástrojů potřebných k úspěšné implementaci. Vybrali jsme platformu knihovny, knihovnu pro práci s neuronovými sítěmi a velkou škálu nástrojů pro přípravu dat.

V dalších kapitolách se zaměříme na samotné použití nástrojů a jejich tvorbu. Ale především budeme diskutovat oba dva způsoby implementace, které nakonec porovnáme.



# Nástroje

Přípravu a zpracování dat si můžeme usnadnit pomocí několika nástrojů. Zaměříme se na hromadné stažení dat ze serveru <http://athena.pef.czu.cz/ptacionline/> a jejich „pročístění“. Dále si představíme nástroje, ve kterých obohatíme stažené snímky o informace potřebné k trénování neuronové sítě.

## 4.1 Hromadné stažení dat

### 4.1.1 Získání dat

Abychom nemuseli stahovat snímky jeden po druhém manuálně, pomůžeme si napsáním jednoduchého skriptu, který stáhne všechny snímky za nás. K tomu nám postačí dva nástroje: `bash` a `wget`. Tento skript stáhne veškerá data, která jsou na serveru <http://athena.pef.czu.cz/ptacionline/> dostupná. Detailní dokumentace skriptu je k nalezení v příloze F).

Algoritmus 4.1: Hromadné stažení dat ze serveru athena.pef.czu.cz.

---

```

1 #!/bin/bash
2
3 DIRECTORY=data
4 URL=http://athena.pef.czu.cz/ptacionline/
5
6 wget -o log.txt -nv --show-progress -c -P "$DIRECTORY" -r -np -nH --cut-dirs=1 -R
      index.html "$URL"

```

---

Stručné vysvětlení příkazu `wget`, který používáme na poslední řádce skriptu 4.1:

- Všechny složky a podsložky dostupné na serveru budou staženy lokálně do složky `$DIRECTORY`.
- `-o log.txt` vytvoří záznam do souboru `log.txt`.
- `-nv` zobrazuje pouze chyby, ne varování.
- `-show-progress` ukáže progres stahování.
- `-c` – pokračuj ve stahování nedokončených souborů.

## 4. NÁSTROJE

---

- **-r** – rekurzivně stahuj podsložky.
- **-np** – nestahuj soubory v složkách výše, než **ptacionline**.
- **-nH** – nestahuj do složky, která se jmenuje stejně jako doména, ale přímo do **\$DIRECTORY**.
- **-cut-dirs=1** – ve složce **\$DIRECTORY** vynech první složku (**ptacionline**).
- **-R index.html** – nestahuj soubory **.html**.

### 4.1.2 Čištění dat

Skript, který jsme představili v kapitole 4.1.1, stáhne veškerá data z daného serveru. Mezi takovými daty jsou snímky, které jsou pro nás užitečné, ale zbytek stažených dat je pro nás zbytečný. Abychom se v datech mohli lépe orientovat a ušetřit místo na pevném disku, bylo by vhodné nepotřebná data smazat. Pro tento účel naprogramujeme jednoduchý nástroj v programovacím jazyce Java, který automaticky ponechá data potřebná a data nepotřebná smaže.

Zdrojový kód a dokumentace nástroje **FolderTrimmer** je k nalezení v příloze D. Výsledný program stačí spustit a složku, která má být promazána, mu předat jako argument: **run.sh /home/demo/eggs/data**.

FolderTrimmer funguje ve dvou režimech. První, základní režim, smaže všechny soubory jiného typu než PNG, TXT a XML. V případě, že složka neobsahuje další složku nebo alespoň jeden soubor typu PNG, TXT nebo XML, bude také smazána. Druhý režim funguje stejně jako první, ale smaže všechny složky, které neobsahují soubor **imgdata.xml**. To umožní vymazání všech dat, které nejsou relevantní pro trénování neuronové sítě. První režim spustíme příkazem **run.sh false "cesta\_ke\_slozce"**. Druhý režim spustíme příkazem **run.sh true "cesta\_ke\_slozce"**.

## 4.2 Příprava trénovacích a testovacích dat

Když máme všechna potřebná data stažená, je potřeba je připravit tak, abychom je mohli použít k trénování neuronové sítě. Příprava dat se liší podle typu implementace, který zvolíme. V případě trénování neuronové sítě k detekci objektů, chceme k jednotlivým snímkům přidat informaci **kde, jaké velikosti a kolik** vajec se v nich nachází. V případě trénování neuronové sítě pro rozpoznávání (klasifikaci) obrazu, je potřeba ke snímkům přidat informaci o **celkovém počtu vajec**.

### 4.2.1 Tagger

Nástroj **Tagger** slouží k manuálnímu označování snímků. Pracuje na úrovni složek, kde jednotlivé složky a snímky reprezentují jednu videosekvenci. Uživatel pak může program spustit a poměrně rychle označit, kolik vajec se na daných snímcích nachází. Tagger je webovou aplikací s jednoduchým uživatelským rozhraním. Uživatel je prezentován všemi snímkům dané složky a má možnost u každého snímků specifikovat počet vajec. Uživateli je složka vybrána automaticky. Po odeslání dat je uživatel vyzván k označení další složky. Tento proces se opakuje do té doby, než jsou označené všechny složky. Ukázka uživatelského rozhraní je vidět na obrázku 4.1.

Výstupem programu jsou soubory **imgdata.xml**, které obsahují informace o počtu vajec na jednotlivých snímcích. Tyto data jsou pak používána pro trénování neuronové sítě k rozpoznávání obrazu, kde počet vajec reprezentuje možné výstupy neuronové sítě. Takto označené snímky

## 4.2. Příprava trénovacích a testovacích dat

---

[Different folder](#)

Prefill the form:

		<input type="text" value="0"/> <input type="button" value="▼"/>
		<input type="text" value="2"/> <input type="button" value="▼"/>
		<input type="text" value="2"/> <input type="button" value="▼"/>
		<input type="text" value="8"/> <input type="button" value="▼"/>
		<input type="text" value="8"/> <input type="button" value="▼"/>

Obrázek 4.1: Tagger – hromadné určování počtu vajec.

## 4. NÁSTROJE

---

se dají použít i pro validaci jakéholidiv řešení – finální softwarové knihovně předáme složku se snímky, knihovna vyhodnotí výsledky a my je poté můžeme porovnat s výsledky, které jsme manuálně nasbírali. Příklad výstupu je přiložen v příloze C.3.

Program je napsán v programovacím jazyce Java. Detailní popis nástroje se nachází v příloze C. Jedná se o webovou aplikaci, která je postavená na technologii Spring Boot [12]. Výpis 4.2 ukazuje algoritmus pro selekci vhodných složek.

Algoritmus 4.2: Výběr vhodných složek pro trénování nástrojem Tagger.

---

```
1 public static ArrayList<String> scanFolder(String location) {
2     File locFile = new File(location);
3
4     if (!locFile.exists()) {
5         return new ArrayList<>();
6     }
7
8     // example folder name: 20160430_073822_526_D
9     final Pattern pattern = Pattern.compile("\\d{8}_\\d{6}_\\d{3}_D");
10    List arr = Arrays.asList(locFile.list((File file, String name) -> {
11        File workingDir = new File(file.getAbsolutePath() + File.separator + name);
12        // needs to be a dir in a correct format
13        if (!workingDir.isDirectory() && !pattern.matcher(name).matches()) {
14            return false;
15        }
16
17        // if already tagged (contains imgdata.xml file)
18        File imgDataFile = new File(workingDir, "imgdata.xml");
19        if (imgDataFile.exists()) {
20            return false;
21        }
22
23        // contains any pictures
24        if (workingDir.list((f, n) -> {
25            File workingFile = new File(f.getAbsolutePath() + File.separator + n);
26            return workingFile.isFile() && FilenameUtils.getExtension(n).equals("png");
27        }).length <= 0) {
28            return false;
29        }
30
31        return true;
32    }));
33
34    return new ArrayList<>(arr);
35 }
```

---

### 4.2.2 LabelImg [13]

Trénovací data pro detekci objektů vyžadují jiný formát než data pro rozpoznávání obrazu. Je potřeba na jednotlivých snímcích označit pozici jednotlivých vajec. Přesně za tímto účelem byl vytvořen nástroj LabelImg [13]. Uživatel si při spuštění programu zvolí složku, ve které má data připravená na označení. Uživatel má možnost zadat typy objektů, které chce na snímcích

## 4.2. Příprava trénovacích a testovacích dat



Obrázek 4.2: Uživatelské rozhraní nástroje LabelImg.

označovat. V našem případě se jedná pouze o jeden typ objektu – vejce<sup>11</sup>. Poté je uživatel prezentován se snímkem, kde má možnost objekty manuálně ohraničit (viz obrázek 4.2).

Níže je přiložen ukázkový výstup programu LabelImg.

```
1 <annotation>
2   <folder>eggs</folder>
3   <filename>snap0001-0000000648.png</filename>
4   <path>C:\Users\test\Desktop\eggs\snap0001-0000000648.png</path>
5   <source>
6     <database>Unknown</database>
7   </source>
8   <size>
9     <width>1280</width>
10    <height>720</height>
11    <depth>3</depth>
12  </size>
13  <segmented>0</segmented>
14  <object>
15    <name>egg</name>
16    <pose>Unspecified</pose>
17    <truncated>0</truncated>
18    <difficult>0</difficult>
19    <bndbox>
20      <xmin>422</xmin>
21      <ymin>323</ymin>
22      <xmax>500</xmax>
23      <ymax>386</ymax>
24    </bndbox>
25  </object>
26  <object>
27    <name>egg</name>
28    <pose>Unspecified</pose>
29    <truncated>0</truncated>
30    <difficult>0</difficult>
31    <bndbox>
32      <xmin>462</xmin>
33      <ymin>379</ymin>
34      <xmax>544</xmax>
35      <ymax>447</ymax>
```

---

<sup>11</sup>Ve skutečnosti se náš objekt (label) jmenuje `egg`.

#### 4. NÁSTROJE

---

```
36      </bndbox>
37  </object>
38 </annotation>
```

---

## **Implementace detekování objektů**

**5.1 Příprava dat**

**5.2 Trénování neuronové sítě**

**5.3 Ověření funkčnosti**



# Implementace rozpoznání obrazu

V této kapitole se zaměříme na implementaci neuronové sítě, která bude mít za cíl rozpoznat, do které kategorie snímek patří. Řešíme tedy problém klasifikace obrazu. Kategorií, do kterých budeme chtít snímky rozřadit, bude **11**:

**Kategorie „0“:** Pro snímky, kde je počet vajec roven **0**.

**Kategorie „1“:** Pro snímky, kde je počet vajec roven **1**.

**Kategorie „2“:** Pro snímky, kde je počet vajec roven **2**.

**Kategorie „3“:** Pro snímky, kde je počet vajec roven **3**.

**Kategorie „4“:** Pro snímky, kde je počet vajec roven **4**.

**Kategorie „5“:** Pro snímky, kde je počet vajec roven **5**.

**Kategorie „6“:** Pro snímky, kde je počet vajec roven **6**.

**Kategorie „7“:** Pro snímky, kde je počet vajec roven **7**.

**Kategorie „8“:** Pro snímky, kde je počet vajec roven **8**.

**Kategorie „9“:** Pro snímky, kde je počet vajec roven **9**.

**Kategorie „10“:** Pro snímky, kde je počet vajec roven **10**.

Celé řešení – příprava dat, trénování neuronové sítě, testování funkčnosti a měření přesnosti – budeme implementovat v programovacím jazyce Python 3.

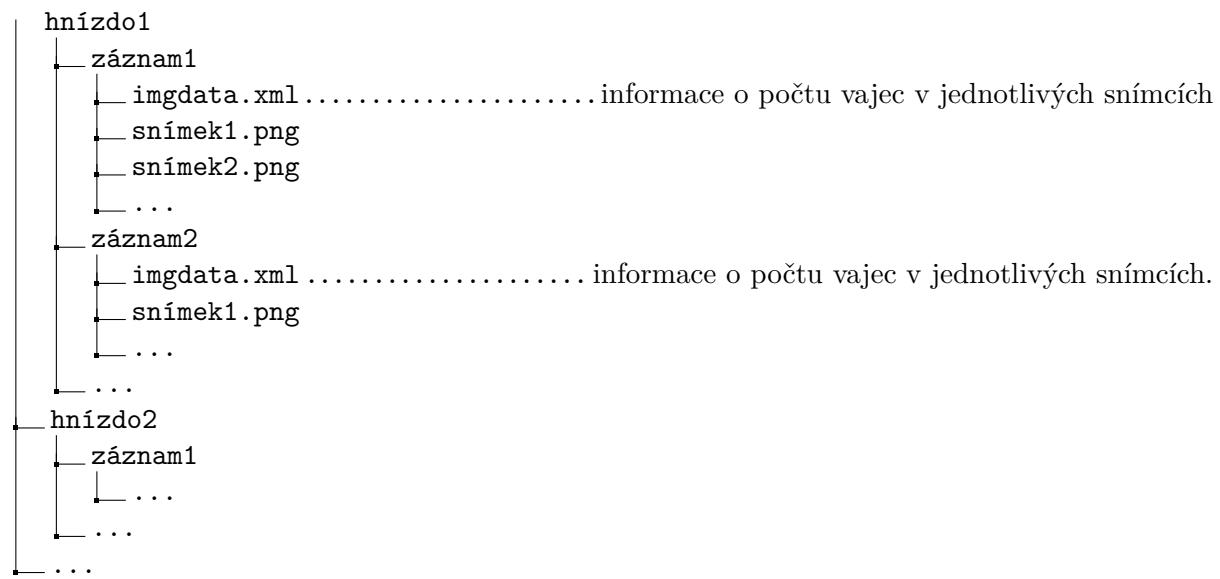
## 6.1 Příprava dat

Abychom nemuseli psát vlastní skripty pro trénování neuronové sítě, ale mohli použít skripty standartně dostupné [14], musíme upravit strukturu našich dat.

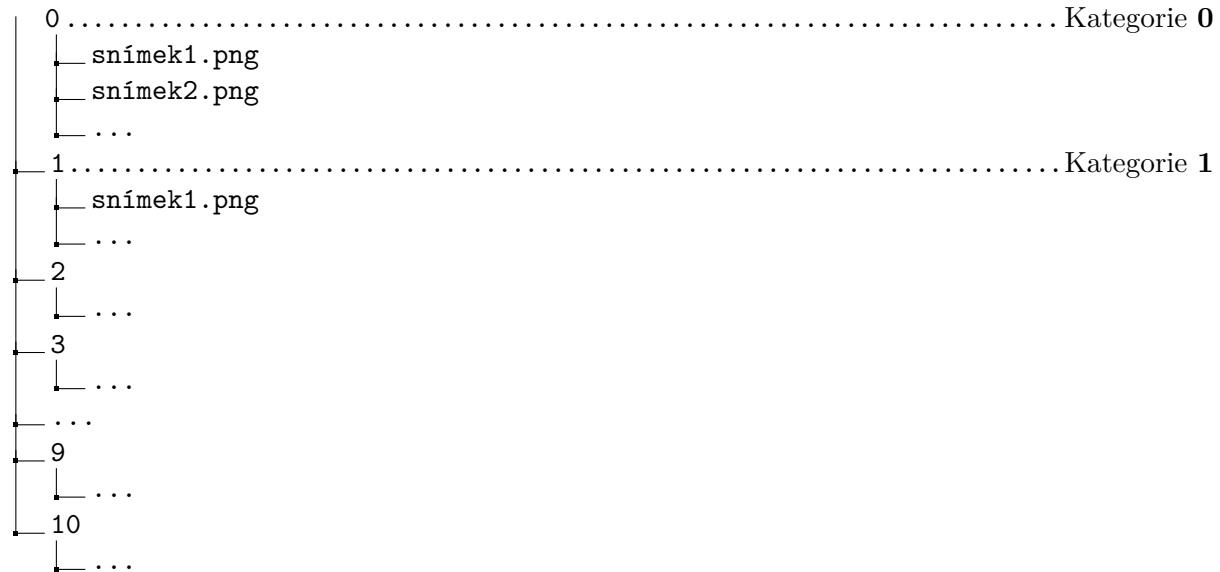
Současná struktura našich dat je následující:

## 6. IMPLEMENTACE ROZPOZNÁVÁNÍ OBRAZU

---



Nová struktura dat, které potřebujeme docílit:



Každá kategorie má vlastní složku. Do každé kategorie patří snímky s počtem vajec, který odpovídá dané kategorii. Jakmile máme naše trénovací data uspořádaná do požadované struktury, je vše připraveno pro trénování neuronové sítě.

### 6.2 Trénování neuronové sítě

Moderní modely pro rozpoznávání obrazu mají miliony parametrů a je **extrémně** výpočetně náročné je vytrénovat. Učení „přenosem modelu“<sup>12</sup> je technika, která ušetří spoustu práce vy-

---

<sup>12</sup>Transfer learning.

užitím již před-trénovaného modelu a přetrénováním pouze finálních vrstev [15]. Více informací k efektivitě tohoto řešení viz kapitola 2.

Předpokladem pro trénování neuronové sítě je nainstalovaná knihovna TensorFlow a všechny její závislosti [16]. Model, ze kterého budeme vycházet je **Inception-v4**<sup>13</sup> [17], který byl vytrénován společností Google na přibližně 1,2 mil. snímků[18]. V soutěži ImageNet [19] drží model Inception-v4 nejlepšího skóre: Top-1 Accuracy<sup>14</sup> 80.2% a Top-5 Accuracy<sup>15</sup> 95.2% [20].

Trénovací data jsou připravena, takže můžeme začít s přetrénováním finální vrstvy našeho modelu:

---

```
1 python3 train_image_classifier.py \
2   --train_dir=egg_recognition/models \
3   --dataset_dir=egg_recognition/training_data \
4   --dataset_name=eggs \
5   --dataset_split_name=train \
6   --model_name=inception_v4 \
7   --checkpoint_path=egg_recognition/inception_checkpoint/inception_v4.ckpt \
8   --checkpoint_exclude_scopes=InceptionV3/Logits,InceptionV3/AuxLogits \
9   --trainable_scopes=InceptionV3/Logits,InceptionV3/AuxLogits
10 python3 tensorflow/examples/image_retraining/retrain.py \
11   --bottleneck_dir=egg_recognition/bottlenecks \
12   --how_many_training_steps=8000 \
13   --model_dir=egg_recognition/models/ \
14   --summaries_dir=egg_recognition/training_summaries/"${ARCHITECTURE}" \
15   --output_graph=egg_recognition/retrained_graph.pb \
16   --output_labels=egg_recognition/retrained_labels.txt \
17   --architecture="${ARCHITECTURE}" \
18   --image_dir=egg_recognition/training_data
```

---

## 6.3 Ověření funkčnosti

info o správných výsledcích

## 6.4 Identifikované problémy

třídy jsou moc podobné (obrázek není zas az tak jiný, jestli tam je jedno nebo dve vajicka) většina části obrazku není pro určení následující relevantní

kdybychom chteli lepsi výsledky, museli bychom nejak predem oriznout obraz, aby vajicka tvorili větinu obrazku

---

<sup>13</sup>Dostupný ke stažení na: [http://download.tensorflow.org/models/inception\\_v4\\_2016\\_09\\_09.tar.gz](http://download.tensorflow.org/models/inception_v4_2016_09_09.tar.gz)

<sup>14</sup>Odpověď modelu (ta s nejvyšší pravděpodobností) přesně odpovídala očekávanému výsledku.

<sup>15</sup>Kterákoli z 5 nejpravděpodobnějších odpovědí modelu odpovídala očekávanému výsledku.



## **Volba řešení**

- 7.1 Srovnání implementací**
- 7.2 Exportování grafu**
- 7.3 Implementace Java knihovny**

Text



## Ověření implementace

- 8.0.1 Metodika vývoje
- 8.0.2 Testování
- 8.0.3 Možnosti vylepšení

Text



---

## **Závěr**



---

## Literatura

- [1] Česká zemědělská univerzita v Praze, Fakulta životního prostředí: *O projektu ptáci online*. [online]. [cit. 2017-12-27]. Dostupné z: <http://www.ptacionline.cz/o-projektu/o-projektu>
- [2] Česká zemědělská univerzita v Praze, Fakulta životního prostředí: *Ptáci online v médiích*. [online]. [cit. 2017-12-27]. Dostupné z: <http://www.ptacionline.cz/o-projektu>
- [3] Oracle: *jar-The Java Archive Tool*. [online]. [cit. 2017-11-14]. Dostupné z: <https://docs.oracle.com/javase/7/docs/technotes/tools/windows/jar.html>
- [4] The Apache Software Foundation: *Maven* [online]. 2017. Dostupné z: <https://maven.apache.org/>
- [5] Ministerstvo životního prostředí: *Ptáci online: Sledujte záběry z „chytré ptačí budky“ na budově MŽP* [online]. [cit. 2017-11-18]. Dostupné z: [http://www.mzp.cz/cz/news\\_160608\\_Ptaci\\_online](http://www.mzp.cz/cz/news_160608_Ptaci_online)
- [6] Šuma, P.: *Detekce počtu mláďat v ptačích hnizdech za pomoci nástrojů rozpoznání obrazu*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.
- [7] Google Inc.: *Image Recognition* [online]. [cit. 2017-11-17]. Dostupné z: [https://www.tensorflow.org/tutorials/image\\_recognition](https://www.tensorflow.org/tutorials/image_recognition)
- [8] Google Inc.: *Tensorflow API Documentation* [online]. [cit. 2017-11-17]. Dostupné z: [https://www.tensorflow.org/api\\_docs/](https://www.tensorflow.org/api_docs/)
- [9] Google Inc.: *Getting Started with TensorFlow* [online]. [cit. 2017-11-17]. Dostupné z: [https://www.tensorflow.org/get\\_started/](https://www.tensorflow.org/get_started/)
- [10] Google Inc.: *Tensorflow Object Detection API* [online]. [cit. 2017-11-17]. Dostupné z: [https://github.com/tensorflow/models/tree/master/research/object\\_detection](https://github.com/tensorflow/models/tree/master/research/object_detection)
- [11] Google Inc.: *TensorFlow: Importing Data* [online]. [cit. 2017-11-17]. Dostupné z: [https://www.tensorflow.org/programmers\\_guide/datasets](https://www.tensorflow.org/programmers_guide/datasets)
- [12] Pivotal Software, Inc.: *Spring Boot* [online]. 2018. Dostupné z: <https://projects.spring.io/spring-boot/>

## LITERATURA

---

- [13] Tzutalin: *LabelImg [online]*. Git code (2015). Dostupné z: <https://github.com/tzutalin/labelImg>
- [14] Google Inc.: *Simple transfer learning with Inception v3 or Mobilenet models. [online]*. [cit. 2017-12-03]. Dostupné z: [https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/image\\_retraining/retrain.py](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/image_retraining/retrain.py)
- [15] Donahue, J.; Jia, Y.; Vinyals, O.; aj.: DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. Technická zpráva, UC Berkeley & ICSI, 2013, [cit. 2017-12-08]. Dostupné z: <https://arxiv.org/pdf/1310.1531v1.pdf>
- [16] Google Inc.: *Installing TensorFlow [online]*. [cit. 2017-12-06]. Dostupné z: <https://www.tensorflow.org/install/>
- [17] Szegedy, C.; Ioffe, S.; Vanhoucke, V.; aj.: Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. Technická zpráva, Google Inc., 2016, [cit. 2017-12-09]. Dostupné z: <https://arxiv.org/pdf/1512.00567.pdf>
- [18] Google Inc.: *v4trainingsize [online]*. [cit. 2017-12-12]. Dostupné z: <https://github.com/tensorflow/models/tree/master/research/slim#preparing-the-datasets>
- [19] Stanford Vision Lab, Stanford University, Princeton University: *ImageNet [online]*. [cit. 2017-12-06]. Dostupné z: <http://www.image-net.org/>
- [20] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, ročník 115, č. 3, 2015: s. 211–252, doi: 10.1007/s11263-015-0816-y.
- [21] Oracle: *Javadoc tool. [online]*. [cit. 2017-11-12]. Dostupné z: <http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html>
- [22] *TexDoclet [online]*. 2017. Dostupné z: <http://doclet.github.io/>

## Seznam použitých zkratek

**GUI** Graphical user interface

**XML** Extensible markup language

**HTML** HyperText Markup Language

**RAM** Random-access memory

**PDF** Portable Document Format

**NN** Neural Network

**CNN** Convolutional Neural Network

**RCNN** Recursive Convolutional Neural Network

**API** Application Programming Interface

**PNG** Portable Network Graphics

**JPEG** Joint Photographic Experts Group



---

# Dokumentace API knihovny

Příloha obsahuje dokumentaci API Java knihovny, která je výsledkem této práce. Dokumentace byla vygenerována ze zdrojových kódů nástroji Javadoc[21] a TexDoclet[22]. Kód je psán v anglickém jazyce, stejně jako jeho dokumentace. Proto je text i zde v angličtině.

## B.1 Package org.cvut.havluja1.eggdetector

<i>Package Contents</i>	<i>Page</i>
Classes	
EggDetector .....	51
SequenceClassifier .....	55

### B.1.1 Class EggDetector

#### B.1.1.1 Count the number of eggs in given images

The egg detector is a library that helps you count the number of eggs in a given folder. Egg detector works by using TeonsorFlow Object Detection API in the background. To learn more, see <https://www.tensorflow.com>.

Example usage:

---

```
1 EggDetector eggDetector = new EggDetector();
2 SequenceClassifier sequenceClassifier = eggDetector.evaluate(new File("image_dir"));
3 System.out.println("final count: " + sequenceClassifier.getFinalCount());
4 System.out.println("individual scores: " + sequenceClassifier.getIndividualCounts());
5 eggDetector.closeSession();
```

---

## B. DOKUMENTACE API KNIHOVNY

---

### B.1.1.2 Declaration

```
1 public class EggDetector  
2 extends java.lang.Object
```

### B.1.1.3 Constructor summary

*EggDetector()*

Constructor loads the pre-trained frozen graph into memory.

### B.1.1.4 Method summary

*closeSession()*

Closes the EggDetector session.

*evaluate(File)*

Runs egg detection on a given *dir*.

*getMinimalConfidence()*

Get the minimalConfidence setting for this instance.

*isDebugMode()*

Get this instance's debug mode setting.

*setDebugMode(boolean)*

Set this instance's debug mode setting.

*setMinimalConfidence(float)*

Set the minimalConfidence setting for this instance.

*toString()*

### B.1.1.5 Constructors

- *EggDetector*

```
1 public EggDetector()
```

– Description

Constructor loads the pre-trained frozen graph into memory.

It also checks whether TensorFlow is supported on your platform.

### B.1.1.6 Methods

- *closeSession*

```
1 public void closeSession() throws java.lang.IllegalStateException
```

- **Description**  
Closes the EggDetector session. This instance of EggDetector will not be usable again.
- **Throws**
  - \* `java.lang.IllegalStateException` – if the session has been closed already by calling `closeSession()`
- *evaluate*

```
1 public SequenceClassifier evaluate(java.io.File dir) throws  
    java.lang.IllegalArgumentException, java.lang.IllegalStateException
```

- **Description**  
Runs egg detection on a given `dir`.
- **Parameters**
  - \* `dir` – a directory containing .jpg or .png files for object detection
- **Returns** –
- **Throws**
  - \* `java.lang.IllegalArgumentException` – if `dir` is not a directory or contains no images
  - \* `java.lang.IllegalStateException` – if the session has been closed already by calling `closeSession()`
- *getMinimalConfidence*

```
1 public float getMinimalConfidence()
```

- **Description**  
Get the `minimalConfidence` setting for this instance.  
Minimal Confidence score is used as a confidence boundary during the process of object detection. An object that has been detected with a confidence score lower than `minimalConfidence` is ignored. An object that has been detected with a confidence score higher or equal than `minimalConfidence` is added to the final result list.
- **Returns** – This instance's `minimalConfidence` setting.

- *isDebugEnabled*

```
1 public boolean isDebugEnabled()
```

- **Description**  
Get this instance's debug mode setting.  
If debug mode is enabled (set to true), the library will open a `JFrame` for each processed image with detections graphically highlighted.

## B. DOKUMENTACE API KNIHOVNY

---

- Returns – debug mode setting for this instance
- *setDebugMode*

```
1 public void setDebugMode(boolean debugMode) throws  
    java.lang.IllegalStateException
```

- Description

Set this instance's debug mode setting.  
If debug mode is enabled (set to true), the library will open a `JFrame` for each processed image with detections graphically highlighted.
  - Parameters
    - \* `debugMode` – turn the debug mode on or off
  - Throws
    - \* `java.lang.IllegalStateException` – if the session has been closed already by calling `closeSession()`
- *setMinimalConfidence*

```
1 public void setMinimalConfidence(float minimalConfidence) throws  
    java.lang.IllegalStateException
```

- Description

Set the `minimalConfidence` setting for this instance.  
Minimal Confidence score is used as a confidence boundary during the process of object detection. An object that has been detected with a confidence score lower than `minimalConfidence` is ignored. An object that has been detected with a confidence score higher or equal than `minimalConfidence` is added to the final result list.
  - Parameters
    - \* `minimalConfidence` – `minimalConfidence` for this instance
  - Throws
    - \* `java.lang.IllegalStateException` – if the session has been closed already by calling `closeSession()`
- *toString*

```
1 public java.lang.String toString()
```

### B.1.2 Class SequenceClassifier

#### B.1.2.1 A class containing object detection results for a given directory

SequenceClassifier is a data class containing the results of object detection for a given directory. When constructed, object detection is performed on all images and results are stored in memory.

Example usage:

---

```
1 EggDetector eggDetector = new EggDetector();
2 SequenceClassifier sequenceClassifier = eggDetector.evaluate(new File("image_dir"));
3 System.out.println("final count: " + sequenceClassifier.getFinalCount());
4 System.out.println("individual scores: " + sequenceClassifier.getIndividualCounts());
5 eggDetector.closeSession();
```

---

#### B.1.2.2 Declaration

```
1 public class SequenceClassifier
2     extends java.lang.Object
```

#### B.1.2.3 Method summary

*getFinalCount()*

Get the final score for the entire directory.

*getIndividualCounts()*

Gets the individual egg count for every image provided.

#### B.1.2.4 Methods

- *getFinalCount*

```
1 public java.lang.Integer getFinalCount()
```

– Description

Get the final score for the entire directory.

The final score is calculated as follows:

- \* individual scores of images are sorted and counted
- \* the highest egg count is returned as a result if we detected this egg count in at least two different images
- \* if no two images contain the same egg count, the highest detected egg count is returned
- \* if no eggs are detected in any of the images, 0 is returned

– Returns – final egg count for this instance

- *getIndividualCounts*

```
1 public java.util.Map getIndividualCounts()
```

## B. DOKUMENTACE API KNIHOVNY

---

- **Description**  
Gets the individual egg count for every image provided.
- **Returns** – A map of individual scores. The key is the filename. The value is the egg count.

## Tagger

Tato příloha obsahuje zdrojový kód a způsob použití nástroje Tagger. Kompletní dokumentace a zdrojový kód je dostupný na přiloženém CD (viz příloha F).

### C.1 Dokumentace (v AJ)

Data tagger is a simple web application used for data tagging used to train a neural network. The data strictly needs to follow the structure of <http://athena.pef.czu.cz/ptacionline/134572snaps/>

#### C.1.1 Usage manual

- Compile (if needed) the app with `mvn clean package`
- Run with `target/run.sh` on Linux or `target\run.bat` on Windows. Both scripts accept data location as the first and only parameter.
- For example: `target/run.sh /home/test/egg/data/nest1`

### C.2 Zdrojový kód

Zdrojový kód je dostupný na přiloženém CD (viz příloha F).

### C.3 Ukázkový výstup

Po určení počtu vajec v každém snímku je do příslušné složky vygenerován soubor `imgdata.xml`, který je použit pro trénování a později testování neuronové sítě. Níže je přiložen ukázkový výpis souboru.

---

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <java version="1.8.0_121" class="java.beans.XMLDecoder">
3   <object class="org.cvut.havluja1.tagger.model.FolderData">
4     <void property="imgData">
5       <object class="java.util.ArrayList">
6         <void method="add">
7           <object class="org.cvut.havluja1.tagger.model.ImgData">
8             <void property="eggCount">
9               <int>0</int>
10            </void>
```

## C. TAGGER

---

```
11     <void property="name">
12         <string>snap0001-0000000668.png</string>
13     </void>
14 </object>
15 </void>
16 <void method="add">
17     <object class="org.cvut.havlujai.tagger.model.ImgData">
18         <void property="eggCount">
19             <int>2</int>
20         </void>
21         <void property="name">
22             <string>snap0002-0000001419.png</string>
23         </void>
24     </object>
25 </void>
26 <void method="add">
27     <object class="org.cvut.havlujai.tagger.model.ImgData">
28         <void property="eggCount">
29             <int>2</int>
30         </void>
31         <void property="name">
32             <string>snap0003-0000002341.png</string>
33         </void>
34     </object>
35 </void>
36 <void method="add">
37     <object class="org.cvut.havlujai.tagger.model.ImgData">
38         <void property="eggCount">
39             <int>8</int>
40         </void>
41         <void property="name">
42             <string>snap0004-0000003160.png</string>
43         </void>
44     </object>
45 </void>
46 <void method="add">
47     <object class="org.cvut.havlujai.tagger.model.ImgData">
48         <void property="eggCount">
49             <int>8</int>
50         </void>
51         <void property="name">
52             <string>snap0005-0000004082.png</string>
53         </void>
54     </object>
55     </void>
56 </object>
57 </void>
58 </object>
59 </java>
```

---

---

## Folder Trimmer

Tato příloha obsahuje zdrojový kód a způsob použití nástroje **FolderTrimmer**. Kompletní dokumentace a zdrojový kód je dostupný na přiloženém CD (viz příloha F).

### D.1 Dokumentace (v AJ)

Folder trimmer is a command line tool for deleting useless data the download script downloads.

#### D.1.1 What does it do

- Deletes files that do not end on either .png, .xml, or .txt.
- If a folder does not contain another folder or at least one of the files listed above, the folder is deleted.

#### D.1.2 Usage manual

- Compile (if needed) the app with `mvn clean package`.
- Run with `target/run.sh` on Linux or `target\run.bat` on Windows. Both scripts accept two parameters:
  - First parameter can be `true` or `false` and it determines whether you want to delete everything except manually tagged data (containing `imgdata.xml`) in case of `true` or if you just want to delete the useless data in case of `false`.
  - Second parameter specifies the root folder of the data that should be trimmed.
  - For example to delete only useless data but keep all the folders with some image data, use: `target\run.bat false C:\school\bakalarka\data`.
  - To delete everything except tagged data, run: `target\run.bat true C:\school\bakalarka\data`.

### D.2 Zdrojový kód

#### D. FOLDER TRIMMER

Algoritmus D.1: Zdrojový kód nástroje FolderTrimmer

```
51         shouldBeDeleted[0] = false;
52         return false;
53     } else {
54         return true;
55     }
56 }
57 }
58
59     return true;
60 });
61
62 if (shouldBeDeleted[0]) {
63     try {
64         FileUtils.deleteDirectory(dir);
65         System.out.println("[D] deleting dir: " + dir.getAbsolutePath());
66     } catch (IOException e) {
67         e.printStackTrace();
68     }
69 } else {
70     if (toBeProcessed.length > 0) {
71         for (File currFile : toBeProcessed) {
72             // if file -> delete
73             if (currFile.isFile()) {
74                 if (currFile.delete()) {
75                     System.out.println("[F] deleting file: " +
76                         currFile.getAbsolutePath());
77                 }
78                 continue;
79             }
80             // if dir -> recursive call
81             if (currFile.isDirectory()) {
82                 findAndDeleteEmptyDirs(currFile);
83             }
84         }
85     }
86 }
87 }
88 }
```

---



## Použité programy

**TexStudio** psaní bakalářské práce v L<sup>A</sup>T<sub>E</sub>Xa překlad do PDF

**Notepad++** úprava textových souborů

**gedit** úprava textových souborů

**IntelliJ IDEA Ultimate** psaní implementace v jazyce Java

**GIT** verzování kódu

**JDK8** komplikace, ladění a testování Java knihovny

**Python 3** zpracování dat, trénování neuronových sítí

**Tensorflow** softwarová knihovna pro strojové učení

**LabelImg** označování trénovacích dat

**javadoc** generování dokumentace do HTML

**TexDoclet** generování dokumentace do L<sup>A</sup>T<sub>E</sub>X

**wget** hromadné stahování dat

**bash** pomocné skripty pro zautomatizování práce



---

## Obsah přiloženého CD

readme.txt .....	stručný popis obsahu CD
bin .....	adresář se spustitelnou formou implementace
doc .....	adresář s dokumentací implementace
src	
impl .....	zdrojové kódy implementace
eggdetector .....	zdrojové kódy implementace knihovny
nn_training .....	zdrojové kódy pro trénování neuronové sítě
tools .....	zdrojové kódy pomocných nástrojů k tvorbě bakalářské práce
thesis .....	zdrojová forma práce ve formátu L <sup>A</sup> T <sub>E</sub> X
text .....	text práce
thesis.pdf .....	text práce ve formátu PDF