



## ZADÁNÍ BAKALÁ SKÉ PRÁCE

**Název:** Detekce objekt v pta ích hnízdech pomocí neuronových sítí  
**Student:** Jan Havl j  
**Vedoucí:** Ing. Josef Pavlí ek, Ph.D.  
**Studijní program:** Informatika  
**Studijní obor:** Softwarové inženýrství  
**Katedra:** Katedra softwarového inženýrství  
**Platnost zadání:** Do konce letního semestru 2017/18

### Pokyny pro vypracování

Navhnete a implementujete knihovnu umož ůující zpracovávat data z kamery umíst ěné v hnízdu (pta í budce) s cílem rozpoznat po et vajec v hnízdu . Pro rozpoznání použijte existující algoritmy um ělé inteligence využívající data získaná ze serveru PtaciOnline.cz a další sw. nástroje projektu BirdObserver (athena.pef.czu.cz).

Postupujte v t ěchto krocích:

1. Prove te detailní specifikaci požadavk ů.
2. Seznamte se s projektem BirdObserver a strukturou dat na serveru PtaciOnline.cz.
3. Prove te analýzu a návrh knihovny.
4. Návrh implementujte, zdokumentujte a vhodným zp ůsobem otestujte.
5. Knihovnu navrhnete a implementujete 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 Tvrdí k, CSc.  
d ěkan

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 Pavlíček, Ph.D.

31. prosince 2017



---

## Poděkování

Chtěl bych poděkovat vedoucímu mé bakalářské práce, panu Ing. Josefovi Pavlíčkovi 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 31. prosince 2017

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2017 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 ptáčích hnízdech pomocí neuronových sítí*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017. Dostupný také z WWW: `<todo>`.



---

# Abstrakt

Tato práce se zabývá návrhem a tvorbou softwarové knihovny pro detekci objektů v obraze 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 data . . . . .	21
3.3 Současný způsob zpracování dat . . . . .	22
3.4 Nefunkční požadavky . . . . .	23
3.5 Funkční požadavky . . . . .	23
3.6 Volba technologií . . . . .	25
3.7 Návrh řešení . . . . .	25
3.8 Shrnutí kapitoly . . . . .	25
<b>4 Implementace detekování objektů</b>	<b>27</b>
4.1 Volba technologií . . . . .	27
4.2 Trénovací data . . . . .	27
4.3 Nástroje . . . . .	27
4.4 Příprava dat . . . . .	27
4.5 Trénování neuronové sítě . . . . .	27
4.6 Ověření funkčnosti . . . . .	27

<b>5 Implementace rozpoznávání obrazu</b>	<b>29</b>
5.1 Volba technologií . . . . .	29
5.2 Trénovací data . . . . .	29
5.3 Nástroje . . . . .	29
5.4 Příprava dat . . . . .	29
5.5 Trénování neuronové sítě . . . . .	29
5.6 Ověření funkčnosti . . . . .	29
<b>6 Volba řešení</b>	<b>31</b>
6.1 Srovnání implementací . . . . .	31
6.2 Exportování grafu . . . . .	31
6.3 Implementace Java knihovny . . . . .	31
<b>7 Ověření implementace</b>	<b>33</b>
<b>Závěr</b>	<b>35</b>
<b>Literatura</b>	<b>37</b>
<b>A Seznam použitých zkratk</b>	<b>39</b>
<b>B Dokumentace API knihovny</b>	<b>41</b>
B.1 Package org.cvut.havluja1.eggdetector . . . . .	41
<b>C Použité programy</b>	<b>47</b>
<b>D Obsah přiloženého CD</b>	<b>49</b>

---

## Seznam obrázků

3.1	Ukázka neupraveného snímku z ptačí budky. . . . .	22
3.2	Vejce nemusí být vždy viditelná. . . . .	22



---

## Seznam algoritmů

3.1	Prvotní návrh uživatelského rozhraní knihovny. . . . .	24
-----	--	----





---

# 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 hnízdě 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 mozkiem 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 hnízdě na jednotlivých snímcích i na sekvencích 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 hnízda.

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.

- 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 každý 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
- 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 současného způsobu sbírání dat, rozbor funkčních a nefunkčních požadavků 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 hnízdicího ptactva. Každá budka obsahuje jednu nebo dvě kamery s nočním přísvitem. Ve vletovém otvoru budky je umístěn pohybový senzor, který spustí natáčení kamer při detekci pohybu. Dále je do budky vestavěn venkovní a vnitřní teplotní senzor, mikrofón a senzor venkovního osvětlení, který regulují funkci přísvitu kamer. 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.



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

Akademičtí pracovníci potřebují nashromáždit velké množství dat, ale taková činnost je velmi časově náročná. Proto jsou na takovou práci najímaní 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 hnízdě. 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 hnízdě [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 hnízdě 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 nejvhodnější 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:

Algoritmus 3.1: Prvotní návrh uživatelského rozhraní knihovny.

---

```
1 // Inicializujeme EggDetector. Knihovna se připraví pro
2 // zpracovávání 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 // síti 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. Například `BufferedImage` a `Graphics2D`. Tato abstrakce nám umožní zpracovat oba dva formáty – JPEG i PNG.

### **3.6 Volba technologií**

### **3.7 Návrh řešení**

#### **3.7.1 Rozpoznávání obrazu**

#### **3.7.2 Detekce objektů**

### **3.8 Shrnutí kapitoly**



## Implementace detekování objektů

- 4.1 Volba technologií
- 4.2 Trénovací data
- 4.3 Nástroje
- 4.4 Příprava dat
- 4.5 Trénování neuronové sítě
- 4.6 Ověření funkčnosti

Text



## **Implementace rozpoznávání obrazu**

- 5.1 Volba technologií**
- 5.2 Trénovací data**
- 5.3 Nástroje**
- 5.4 Příprava dat**
- 5.5 Trénování neuronové sítě**
- 5.6 Ověření funkčnosti**

Text





## Volba řešení

- 6.1 Srovnání implementací
- 6.2 Exportování grafu
- 6.3 Implementace Java knihovny

Text



## Ověření implementace

- 7.0.1 Metodika vývoje
- 7.0.2 Testování
- 7.0.3 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 hnízdech za pomoci nástrojů rozpoznání obrazu*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.
- [7] Oracle: *Javadoc tool*. [online]. [cit. 2017-11-12]. Dostupné z: <http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html>
- [8] *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[7] a TexDoclet[8]. 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 .....	41
SequenceClassifier .....	45

### 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.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.

- *isDebugMode*

```
1 public boolean isDebugMode()
```

- **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.

- **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()
```

- **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.



---

## Použité programy

**TexStudio** psaní bakalářské práce v  $\text{\LaTeX}$ a překlad do PDF

**Notepad++** úprava textových souborů

**gedit** úprava textových souborů

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

**GIT** verzování kódu

**JDK8** kompilace, 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  $\text{\LaTeX}$

**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 $\text{\LaTeX}$
text .....	text práce
thesis.pdf .....	text práce ve formátu PDF