

## Bachelor thesis

# Development and implementation of a mobile App for a picture based animal counter

Submitted by: Hao Yuan


Department: Electrical Engineering and Computer Science

Degree program: Information Technology

First examiner: Prof. Dr. Heeren

Date handing out: 15<sup>th</sup> March 2022

Date handing in: 15<sup>th</sup> June 2022

  
(Prof. Dr. Andreas Hanemann)  
Head of Examination Board

**Task description:**

A mobile App for a picture based animal counter should be developed.

After the requirement analysis for the need of having an App for a picture based animal counter, the student has to develop and implement a suitable mobile App for the identified requirements.

Therefore it is necessary to discuss and implement a software architecture, which is able to fulfill the requirements of a mobile Application in this area. Furthermore the information handling, especially the handling of countstrategies and picture handling, must guaranteed through useful functions. The set of functions should be identified through the requirement analyzation. The Application must be able to handle the counting process by having it's own set for test data, which must be created by the student. The counting information must be used in combination with the picture for the visualization of the results in a useful manner.

Testing the solution in connection with the identified requirements should also be a part of the thesis with a critical comparison of possible alternative ways for the implementation.



Prof. Dr. Heeren

### **Statement on the bachelor thesis**

I assure you that I wrote the work independently, without outside help.

Only the indicated sources has been used in the preparation of the work.  
The text or the sense of the text are marked as such.

I agree that my work may be published, in particular that the work may be submitted to third parties for inspection or that copies of the work may be made for forwarding to third parties.

07.06.2022  
Date

Hao Yuan  
Signature

Zusammenfassung der Arbeit

Abstract of Thesis

Fachbereich: **Electrical Engineering and Computer Science**

Department:

Studiengang: **Information Technology**

University course:

Thema: **Development and implementation of a mobile App for a**

Subject: **picture based animal counter**

Zusammenfassung:

Abstract :

The following work is a process of developing of a mobile App for a picture based animal count, and the results of image detection will be visually marked out. Convolutional neural network is applied to object detection. Therefore, the picture-based animal counter is based on a training model which has the characteristics of no need of high computing power, but high efficiency and high accuracy. The mobile App has a visual window, which can be used by users to enter the picture and get the species and amount of the animal in this picture.

Verfasser: **Hao Yuan**

Author:

Betreuender Professor/in: **Prof. Dr. Menno Heeren**

Attending Professor:

WS / SS: **SS 2022**

# Table of Contents

<b>Abstract of Thesis.....</b>	<b>i</b>
<b>Table of Contents .....</b>	<b>ii</b>
<b>1. Introduction.....</b>	<b>1</b>
1.1. Motivation .....	1
1.2. Goal .....	1
1.3. Organization .....	1
<b>2. Related work .....</b>	<b>2</b>
2.1. One-stage method.....	2
2.1.1. YOLO - You Only Look Once.....	2
2.1.2. SSD- Single Shot Detection .....	2
2.2. Two-stage method.....	2
2.2.1. R-CNN .....	2
2.2.2. Fast R-CNN.....	2
2.2.3. Faster R-CNN .....	3
<b>3. Method .....</b>	<b>4</b>
3.1. Preliminary work and preprocessing .....	4
3.1.1. Preparation .....	4
3.1.2. Data collection .....	6
3.1.3. Image Annotation.....	8
3.2. Method and process.....	10
3.2.1. Computer Vision (CV).....	10
3.2.2. Convolutional neural network (CNN).....	12
3.2.3. YOLOv5 Algorithm.....	13

3.2.4.	Detectron2 .....	18
3.2.5.	Comparison and application of algorithm .....	20
3.2.6.	Web programming .....	22
3.2.7.	Mobile app on Android .....	27
<b>4.</b>	<b>Evaluation.....</b>	<b>30</b>
4.1.	Validation .....	30
4.1.1.	Several parameters .....	31
4.1.2.	Validation result .....	33
4.2.	Quality test .....	37
4.2.1.	Image size .....	37
4.2.2.	Amount of detections .....	39
4.2.3.	Small objects .....	39
4.2.4.	Occlusion .....	40
4.3.	User test.....	41
4.3.1.	Recruit testers.....	41
4.3.2.	Scenario Design .....	42
4.3.3.	Test Result.....	42
4.4.	Improvement on App.....	44
4.5.	Limitation .....	45
<b>5.</b>	<b>Conclusion and outlook.....</b>	<b>47</b>
5.1.	Conclusion.....	47
5.1.1.	Contributions.....	47
5.1.2.	Key findings.....	47
5.2.	Outlook.....	47
	<b>Acknowledgments.....</b>	<b>49</b>

<b>Appendix A - List of figures .....</b>	<b>50</b>
<b>Appendix B - List of listings .....</b>	<b>51</b>
<b>Appendix C - List of tables .....</b>	<b>51</b>
<b>Bibliography .....</b>	<b>53</b>

# **1. Introduction**

## **1.1. Motivation**

A picture-based search is now very popular. The areas of interactive search with the greatest societal impact have been in image search engines and recommendation systems. Google, Yahoo! and Microsoft have added interactive visual content-based search methods into their worldwide search engines, which allows search by similar shape and/or color and are used by millions of people each day [1]. However, most image-based search engines can only distinguish the main object categories in the image, but cannot count the specific types of objects in the image. Actually, some search engines only provide the closest pictures and provide the source, which is not able to meet specific needs of users. For this reason, a mobile App that can recognize and count animal species in the picture needs to be developed.

## **1.2. Goal**

This thesis aims to introduce the process of compiling app, including collecting data, processing data, apply model, UI design, web programming and analysis.

## **1.3. Organization**

The rest of the document is structured as follows. Chapter 2 explains the related works of object detection. It compares the accuracy and calculation time of the algorithms as well. Chapter 3 describes the methods used, first introduces the preliminary work and experimental environment, and then introduces the image detection algorithms and web development in detail. Chapter 4 explains the evaluation results, including the verification results of the model on the verification set, the quality test and the user test of the app. Chapter 5 summarizes the work, and suggests future work.



## **2. Related work**

In the field of object detection, there are two main methods. The first one is one-stage method and the other is two-stage method.

### **2.1. One-stage method**

#### **2.1.1. YOLO - You Only Look Once**

It is the first one-stage method, provides extremely fast detection speed and accuracy. It has only one layer of network evaluation, so it is hundreds or even thousands of times faster than other object detection algorithms.

In 2017, the developer improved YOLO and published YOLO9000 and one year later, published YOLOv3.

#### **2.1.2. SSD- Single Shot Detection**

Similar to YOLO, SSD only needs one shot to complete the detection. It uses some technologies, such as multi-scale features and default boxes, so that the accuracy can reach almost the same level as fast-R-CNN model.

### **2.2. Two-stage method**

#### **2.2.1. R-CNN**

Rajeshwari claimed that R-CNN divides a scene into 2000 regions which highly reduces the number of regions where detections need to be made. These regions are generated using a selective search algorithm where greedy algorithm is used to recursively combine similar regions into larger regions. Afterwards, the algorithm put forward the final candidate regional proposal, then these areas are input into SVM to classify the existence of objects [2].

#### **2.2.2. Fast R-CNN**

The fast R-CNN model is established to overcome some shortcomings of the previous R-CNN model. In this method, similar to the previous method, selective search is used to generate region suggestions, but the input image is fed to CNN to generate convolution feature map, which is then used to identify regions and combine them into larger squares by using pooling. Finally, the softmax layer is used to predict the category of the proposed area.

### **2.2.3. Faster R-CNN**

In 2015, S. Ren et al. proposed Faster R-CNN detector shortly after the Fast R-CNN. The author claimed that their method enables a unified, deep-learning-based object detection system to run at 5-17 fps [3].

“The main contribution of Faster R-CNN is the introduction of Region Proposal Network (RPN) that enables nearly cost-free region proposals.” [4]

### 3. Method

#### 3.1. Preliminary work and preprocessing

For this experiment, we need to build a development environment. Also, it is necessary to collect enough appropriate data, which needs to be preprocessed before making the dataset.

##### 3.1.1. Preparation

**Python** is a newly developing programming language. It is famous for its easy learning and rapid development. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms [5]. In this experiment, Python is mainly used in programming the collecting data function and the deep learning model. PyCharm is the IDE((Integrated-Drive-Electronics)) mainly used. Downloading PyCharm as IDE and adapting environment variables is the first step in preparation

**JavaScript** is a powerful programming language with plenty of toolkits which is mainly used in this experiment to programming the web function. We can use PyCharm as the IDE for JavaScript and HTML5 as well.

##### 3.1.1.1. *Environment Setting*

Several environment settings for the experiment.

##### 3.1.1.2. *Operating System*

Operating System	Windows 10
Version number	21H1
Date of installation	2021/1/10
System build	19043.1586
Serial number	MP1DTMFC
Experience	Windows Feature Experience Pack 120.2212.4170.0

**Table 3.1 System parameter**

### 3.1.1.3. *Compiler Environment*

Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)] :: Anaconda, Inc.  
on win32

Python package requirements:

PACKAGE	VERSION
MATPLOTLIB	3.2.2
NUMPY	1.18.5
OPENCV-PYTHON	4.1.1
PILLOW	7.1.2
PYYAML	5.3.1
REQUESTS	2.23.0
SCIPY	1.4.1
TORCH	1.7.0
TORCHVISION	0.8.1
TQDM	4.41.0

**Table 3.2 Package requirements**

### 3.1.1.4. *Hardware foundation*

All compilation process, training process, testing process are done by personal computer. The following is the hardware configuration:

CPU	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz
RAM	8.00 GB
GPU	NVIDIA-SMI 460.32.03
(Supported by Google colab)	Driver Version: 460.32.03 CUDA Version: 11.2

**Table 3.3 Hardware parameter**

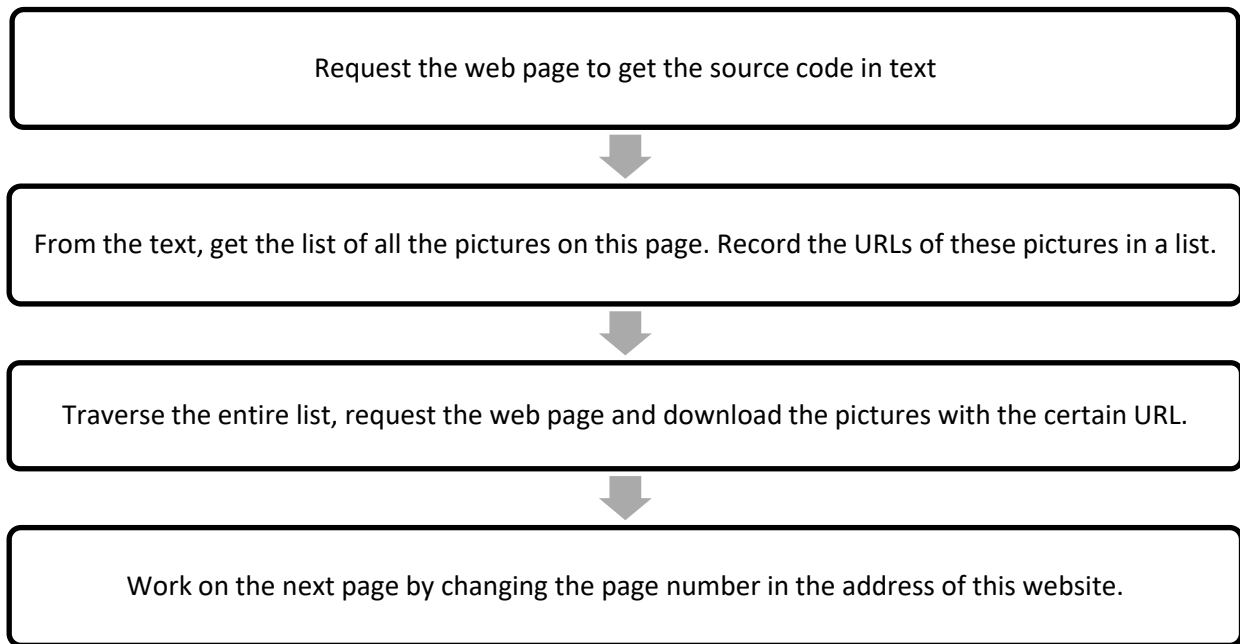
### 3.1.2. Data collection

In order to create a dataset of pictures of different species of animals, the original picture data needs to be clear animal photos. These pictures should have animals features in the picture, and the background should be as concise as possible without interference just like Figure 3.1 shown. The free image website(<https://www.freeimages.com/>) provide a great quantity of clear pictures.



**Figure 3.1** Example of a original image – dog image

Using the python code to collect these images from the website takes much less time than manual download. The request package is used to automatically request the web page. After getting the original code of the web page, use the method of intercepting the string to get the address of the picture. Then save these pictures on the hard disk. In this way, all the pictures can be downloaded efficiently and only the **Uniform Resource Locator (URL)** of the website is needed. Figure 3.2 reveals the logic of the image collection code.



**Figure 3.2 Logic diagram for automatically downloading web page pictures**

Although some images on free image net meet the requirements, more types of animal images are needed. Google pictures is used as a second image library. However, the design of Google pictures web page is different from that of free image web page. It uses neither JavaScript nor page feed to update requests. Therefore, another fully automatic image download function needs to be implemented.

In this case, selenium package from python is used to do the automated test on web driver. Selenium is a tool for testing web applications. Compared with other testing tools, the biggest benefits of using selenium are as follows. Selenium tests run directly in the browser just as real users do. Selenium tests can be run in Internet Explorer, chrome, and Firefox on windows, Linux, and Macintosh. No other testing tool can cover so many platforms. There are many other benefits to using selenium and running tests in a browser.

The browser in this experiment is Google Chrome version 100.0.4896.127( 64bit ) and the web driver is Chrome Driver 100.0.4896.60. The core logic of the code is as follows: Firstly, visit the website with Chrome Driver. Request the current web page source code, find the picture link and save it to the local folder. Then through the automatic control statement to realize the

automatic sliding of the wheel, pull the web page interface, and make the server load more pictures. At this time, more pictures are available.

```
def init_browser(self):
    create browser, send arguments
    browser = webdriver.Chrome(chrome_options=chrome_options)
    browser.get(self.url)
    return browser

def download_images(self, browser, round=2):
    create dir(..) # create dir for images
    for i in range(round):
        img_elements = browser.find_elements_by_tag_name('img') #
find images
        for cookie in cookies: # add cookies to browser
            browser.add_cookie(cookie)
        browser.refresh()
        for img_element in img_elements:
            try:
                download this image
            except:
                print('failure')
```

**Listing 3.1 Get images from Google by selenium**

Through these two methods, a total of 203 pictures of 8 different animals from image net and Google image library are collected. Artificially searched these pictures and deleted unreasonable pictures, such as animated characters, thumbnails, nonanimal pictures etc., the final number of documents is 150. These files are placed in one folder and used as validation sets. At the same time, 10 images selected manually is used as the test dataset.

### 3.1.3. Image Annotation

It is necessary to label these pictures for validation and test. In this experiment roboflow web app(<https://app.roboflow.com/>) is used to mark the collected animal pictures. The method of image marking is shown in the Figure 3.3. Use the rectangular box selection tool to mark the outline of the animal and give it the correct label name, such as bear, cat and dog.

A total of 150 images were manually labeled, including 324 labels. There are eight kinds of animal labels, namely cat, dog, bear, horse, elephant, bird, cow and sheep. Thumbnails of some tagged animal pictures are shown in the Figure 3.4.

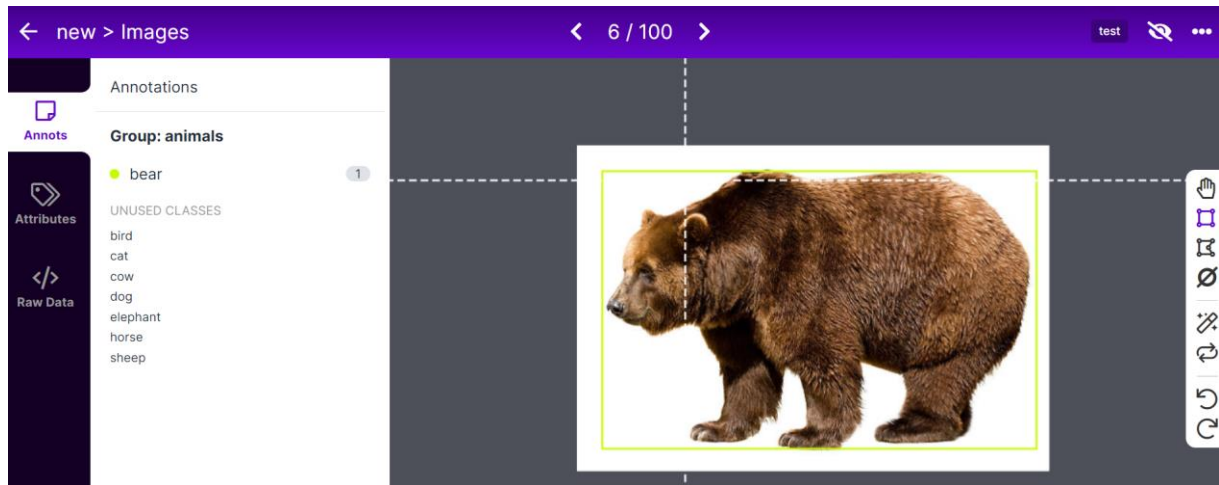


Figure 3.3 Annotating Image by roboflow

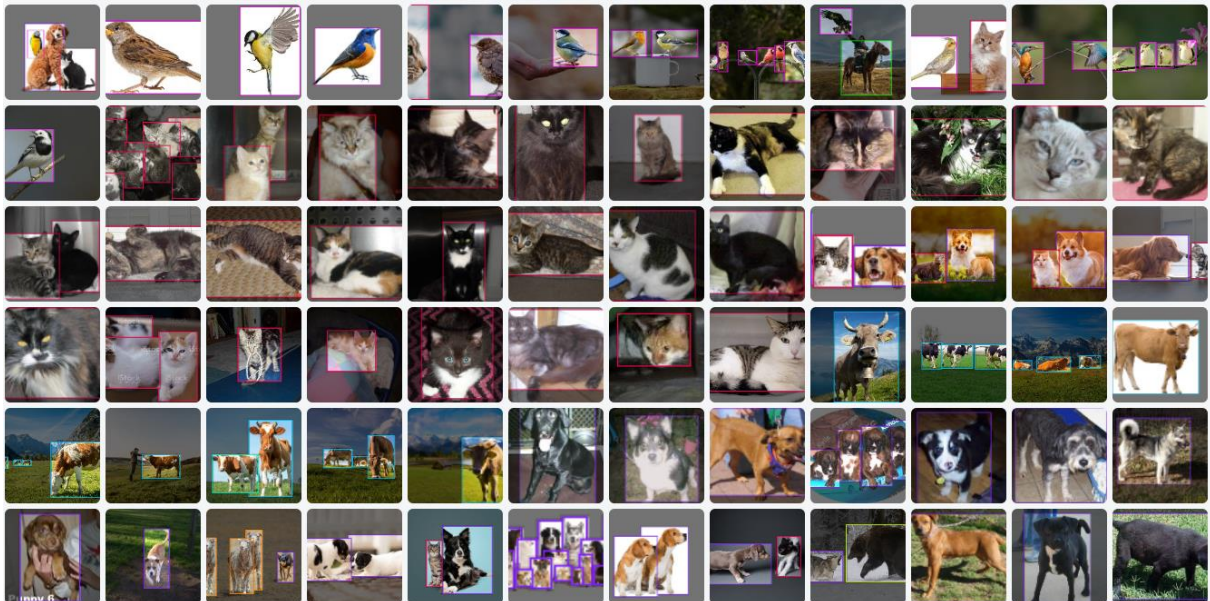


Figure 3.4 Thumbnails of some tagged animal pictures



## **3.2. Method and process**

The goal of this experiment can be considered as an application of object detection.

### **3.2.1. Computer Vision (CV)**

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos [6].

The principle of human vision is as follows: the pupil takes in pixels to obtain the original information, then the cerebral cortex makes preliminary processing, then abstracts some shape features, and then further abstracts some local features, so as to make judgment and identify specific object categories.

Similar to the process of human analyzing pictures, computer vision processes pictures through the following process: constructing multi-layer neural networks, identifying primary image features at the lower level, forming higher-level features with several lower level features, and finally making classification at the top level through the combination of multiple levels.

#### **3.2.1.1. *Challenges on computer vision***

Computer vision has two major challenges.

##### **3.2.1.1.1. *Difficult to extract features***

The pixel difference of the same animal is very large under different angles, different lights and different actions. Even for the same photo, the pixel difference is very large after rotating 90 degrees. In addition, the position of an object in the picture conforms to the perspective principle, so its size is related to the distance from the camera lens.

Therefore, the content in the picture is similar or even the same, but at the pixel level, it will change largely. This is a big challenge for feature extraction.

##### **3.2.1.1.2. *Large amount of data to be processed***

For a 640pixels \* 640pixels color picture, it has RGB three channels, each channel has 640 pixels  $\times$  640 pixels, a total of 122880 parameters, each parameter value is between 0 and 255, which is a very large amount of data. Even if such a color picture is not large for current photography technology and hard disk technology, the amount of data is huge for computer vision trying to understand it.

### **3.2.1.2. Object detection**

Computer vision includes eight tasks. Object detection is one of them. The goal of object detection task is to give an image, let the computer find out the location of all targets, and give the specific category of each target.

Object detection can be applied in many different fields, including medical, academic, biological, anthropological, etc.

#### **3.2.1.2.1. Multi-scale in object detection task**

The size of the input image has a significant impact on the performance of the detection model. In fact, multi-scale is one of the most obvious techniques to improve the accuracy. In the basic network, a feature map ten times smaller than the original image is often generated, which makes the feature description of small objects difficult to be captured by the detection network. By inputting larger and more size images for training, the robustness of the detection model to the size of objects can be improved to a certain extent. Only introducing multi-scale in the test phase can also enjoy the gain brought by large size and multi-scale.

#### **3.2.1.2.2. Detection on small objects**

Because large-scale objects have large area and rich features, their semantic information will appear in the deeper feature map, which is generally easier to detect

Small objects (usually considered to be less than  $32 \times 32$  in absolute size can be regarded as a small object or the object width and height are less than  $1/10$  of the original image width and height, and can be regarded as a small object). Due to its small size, the available features are limited, and its semantic information appears in the shallow feature map. With the deepening of the network, its detailed information may disappear completely.

In brief, detection of small objects is one of the biggest challenges in the field of object detection.

#### **3.2.1.2.3. Border regression technique**

The bounding box regression is a very important technology for object detection. Its purpose is to further improve the position of the modified prediction box according to the initial anchor box. Since the development of object detection for decades, the current object detection algorithms have basically realized the mode from feature to BB.

#### **3.2.1.2.4. Challenges on object detection**

The size of the target to be detected is very small, resulting in a small proportion and great difficulty in detection.

The scale of the target to be detected changes greatly, and the network is difficult to extract efficient features.

The background of the target to be detected is complex, the noise interference is serious, and the detection is difficult.

The contrast between the target to be detected and the background color is low, so it is difficult to extract the discriminative features from the network.

The number of objects to be tested is extremely unbalanced, resulting in sample imbalance.

It is difficult to achieve a good balance between the speed and accuracy of the detection algorithm.

### **3.2.2. Convolutional neural network (CNN)**

If the traditional fully connected neural network is used to process large-scale images, it has three obvious disadvantages:

- (1) Firstly, the spatial information will be lost when the image is expanded into a vector;
- (2) Secondly, too many parameters lead to low efficiency and difficult training;
- (3) At the same time, a large number of parameters will soon lead to over fitting of the network.

In order to solve these problems in the field of computer vision, convolutional neural network is used. CNN can extract the features in the picture and realize the transformation from 3D in reality to 2D in the picture. Different from conventional neural networks, the neurons in each layer of CNN are arranged in three dimensions: width, height and depth.

Convolutional neural network (CNN) is mainly composed of feature extraction layer and feature mapping layer. It can independently extract image features and directly use the original image, avoiding the complex preprocessing process of the input image. Convolutional neural network is composed of a series of convolution layer, activation layer, pooling layer and full connection layer. The convolution process extracts different features of the input signal. Each convolution kernel extracts a single feature on the whole feature map. Multi-core convolution makes the features fully extracted, and the extracted features are transmitted to the next layer as input. These features are progressive from low-level to high-level. Therefore, the deep network structure makes the characteristics of learning more global.

#### **3.2.2.1. Convolution layer**

Convolution layer is the core layer of constructing convolution neural network, which produces most of the computation in the network. The convolution layer acts as a filter. The neural network can learn to activate when it sees some features. The specific visual features may be the boundaries in some directions, or the spots of some colors on the first layer, or even the honeycomb or wheel patterns on the higher layer of the network.

The convolution layer can be regarded as the output of a neuron to reduce the number of parameters and prevent over fitting.

### **3.2.2.2. Pooling**

Usually, a pooling layer is periodically inserted between successive convolution layers. Its function is to gradually reduce the spatial size of the data volume, so as to reduce the number of parameters in the network, reduce the consumption of computing resources, and effectively control over fitting. The convergence layer uses the max operation to operate each depth slice of the input data volume independently to change its spatial size. The most common form is that the convergence layer uses a 2x2 filter to downward sample each depth slice in steps of 2, and 75% of the activation information is lost. Each Max operation takes the maximum value from the four numbers (that is, a 2x2 area in the depth slice), and the depth remains unchanged.

### **3.2.2.3. Famous net of CNN**

Alex Net: Alex net convolutional neural network is popular in the field of computer vision. It is implemented by Alex krizhevsky, Ilya sutskever and Geoff Hinton. Alexnet won the ImageNet ILSRV competition in 2012.

Google Le Net: the winner of ILSVRC 2014 is the convolutional neural network implemented by a team leads by Szeged from Google. Its main contribution is to implement a foundation module, which can significantly reduce the number of parameters in the network (there are 60m in Alex Net and only 4m in this network). In addition, instead of using the full connection layer at the top of the convolutional neural network, this paper uses an average convergence to remove a large number of unimportant parameters.

VGG Net: the second place in ILSVRC 2014, is the convolutional neural network implemented by Karen Simonyan and Andrew Zisserman, which is now called VGG net. Its main contribution is to show that the depth of the network is the key part of the excellent performance of the algorithm. Their best network consists of 16 convolution and full connection layers. The structure of the network is very consistent. 3x3 convolution and 2x2 convergence are used from beginning to end. Their pre training model can be obtained on the network and used in Caffe.

RES NET, residual network, is the winner of ILSVRC 2015, which was implemented by Kaiming He and his teammates. It uses special jump links and a lot of batch normalization. This structure also does not use the full connection layer at the end.

### **3.2.3. YOLOv5 Algorithm**

Redmon, the creator of YOLO algorithm claimed “We introduce YOLO, a unified model for object detection. Our model is simple to construct and can be trained directly on full images.

Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly.” [7]

YOLOv5 is the latest version of YOLO algorithm, which includes a set of models pretrained based on coco data set.

“YOLOv5 is a family of object detection architectures and models pretrained on the COCO dataset, and represents Ultralytics open-source research into future vision AI methods, incorporating lessons learned and best practices evolved over thousands of hours of research and development.” [8]

### **3.2.3.1. Detect model**

There are four versions in the YOLOv5 family, namely YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x. Take the YOLOv5s model as example, the structure of this model can be divided into four parts.

#### **3.2.3.1.1. Input**

MOSAIC is the algorithm for data enhancement. The trained images are randomly scaled, cut and arranged to achieve the effect of feature enhancement. In YOLO algorithm, for different data sets, there will be anchor boxes with initial set length and width. During each training, the optimal anchor frame values in different training sets are calculated adaptively. To deal with images of different sizes, YOLOv5 algorithm uses an adaptive image resizing method to improve the training efficiency by adding the least black edges.

#### **3.2.3.1.2. Backbone**

The original  $608 * 608 * 3$  image is input into the focus structure and sliced into a  $304 * 304 * 12$  feature map. In addition, the algorithm uses CSP net (Cross Stage Partial Network) structure to solve the problem of too much computation. The creator of CSP net believes that the problem of excessive reasoning calculation is caused by the repetition of gradient information in network optimization (Wang et al., 2019). Therefore, the CSP net is used to divide the feature mapping of the basic layer into two parts, and then merge them through the cross-stage hierarchy, which can reduce the amount of calculation and ensure the accuracy.

#### **3.2.3.1.3. neck**

FPN (feature pyramid networks) layer conveys strong semantic features from top to bottom and PAN (Path Aggregation Network) conveys strong positioning features from bottom to top. The two are combined to aggregate parameters of different detection layers from different trunk layers to achieve the purpose of feature extraction.

#### **3.2.3.1.4. prediction**

Prediction: GIoU\_ loss algorithm can solve the problem when the bounding boxes do not coincide on the basis of IOU algorithm.

In this experiment, YOLOv5x6 is used. The YOLOv5x6 model is the newly published version of YOLOv5x. As can be seen in Figure 3.5, compared with other pretrained models, this model has the most depth neural network layers and the longest calculation time. In this experiment, the calculation time is not considered to be the key consideration, and the accuracy is more valuable, so the algorithm is used as the model for predict the species and number of animals. It can also realize the frame selection of animal outline and point out the specific coordinates of animals in the picture.

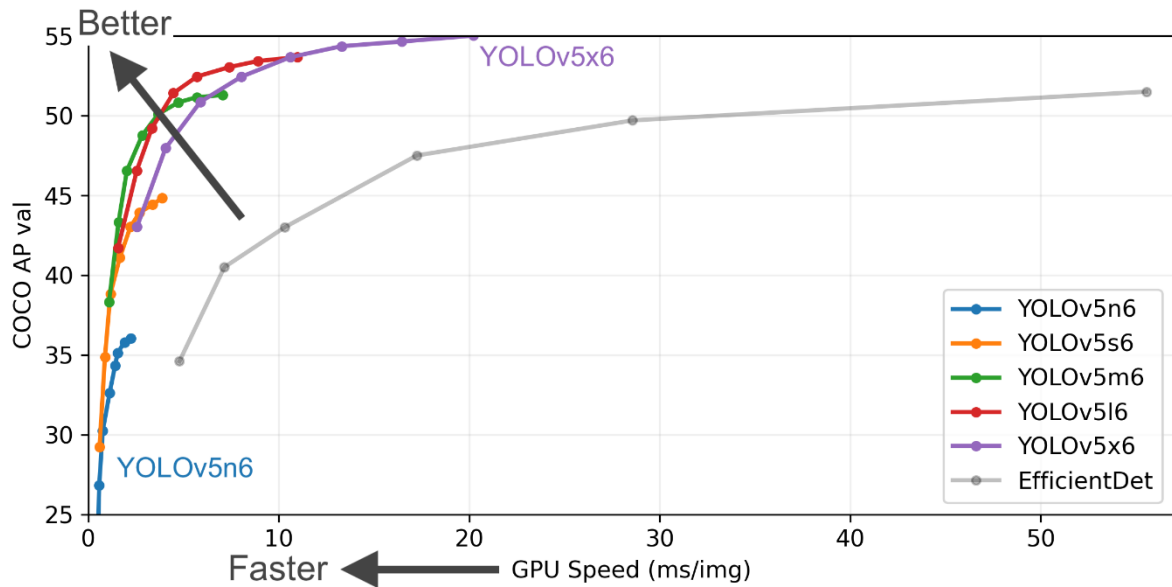


Figure 3.5 The comparison of different versions of YOLOv5 model

One picture predicted and detected by this model are shown in the Figure 3.6. In this picture, the position marked with a box is the automatically detected animal target, the text box above shows the detected animal type, and the number is the prediction probability. The specific contents of the label file are displayed on the right. Each line represents an animal detected. The first data represents the label index of the animal type, and the following four-dimensional arrays are X1, Y1, X2 and Y2 respectively, which are used to determine the coordinates of the

animal in the picture. At the bottom saw the size of this picture, test results and the time of a single detection.

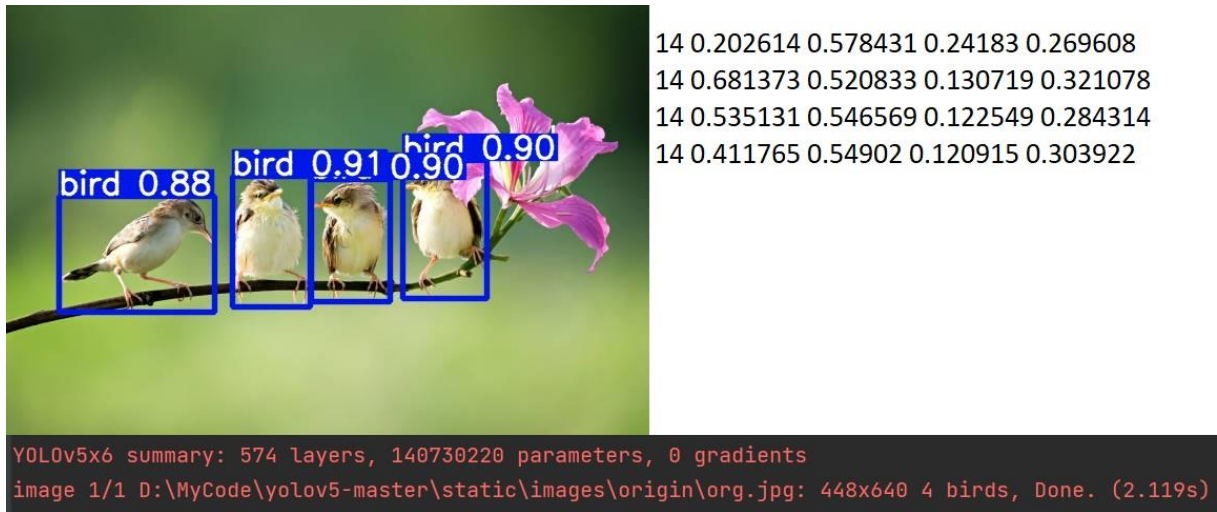


Figure 3.6 The result of detection on a bird image by model

### 3.2.3.2. Dataset

The YOLOv5x6 is pretrained under Microsoft COCO (Common Objects in Context) dataset. It is a large-scale object detection, segmentation, and captioning dataset published by Microsoft and is popular in Machine Learning.

“Understanding visual scenes is a primary goal of computer vision; it involves recognizing what objects are present, localizing the objects in 2D and 3D, determining the object’s attributes, and characterizing the relationship between objects. Therefore, algorithms for object detection and object classification can be trained using the dataset.” [9].

Train dataset includes 118287 images, validation set includes 5000 images and test set includes 20288 images.

### 3.2.3.3. Labels

There are 80 labels in the train dataset, including common items, people and animals. For animal category labels, the following table shows the index and corresponding animal names.

INDEX	NAME
14	Bird
15	Cat
16	Dog

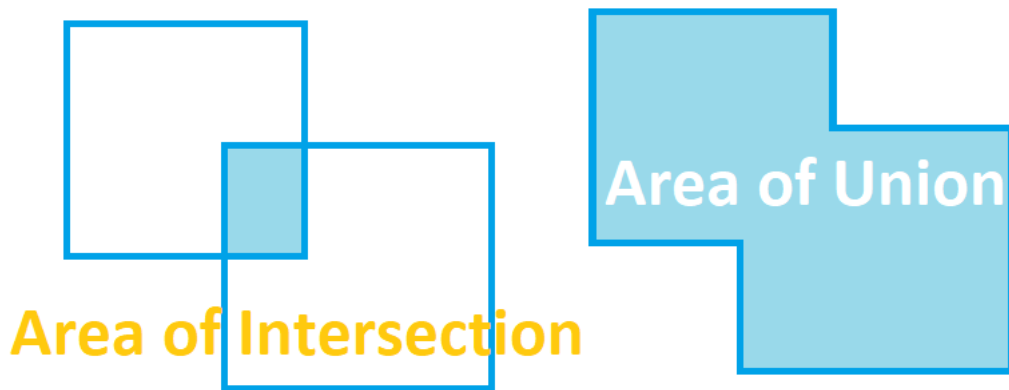
Method	
17	Horse
18	Sheep
19	Cow
20	Elephant
21	Bear

**Table 3.4 Labels of animals in the train dataset**

In this experiment, the model only needs to detect animal labels, so the default value of the class parameter is adjusted to [14,15,16,17,18,19,20,21] at runtime.

### 3.2.3.4. IoU Algorithm

IoU refers to Intersection over Union, which is an algorithm in object detection.  $IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$ . As is shown in Figure 3.7, the area of intersection refers to the area where two predicted bounding box overlap and the area of union refers to the area encompassed by two predicted bounding boxes.



**Figure 3.7 Schematic diagram of IOU algorithm**

In YOLOv5x6 model, the detected object is marked by a rectangular box. In one detection algorithm, multiple rectangular boxes may point to the same area and the prediction results are the same. In this case, IoU algorithm is needed. Setting an appropriate IOU threshold needs to be considered to ensure the higher accuracy of the prediction results of the model. When the threshold is too high, the model tends to regard two adjacent objects as the same object. On the contrary, when the threshold is too low, the model is more inclined to predict a whole into multiple objects.

In this experiment, the accuracy of boundary determination is considered less important than the precision of detection of quantity, so the value of IoU is set to 0.45 by default.



### 3.2.4. Detectron2

Detectron2 is Facebook AI Research's next generation library that provides state-of-the-art detection and segmentation algorithms. It is the successor of Detectron and maskrcnn-benchmark. It supports a number of computer vision research projects and production applications in Facebook [10].

With the detectron2 project, we can test the ability of different models in animal image detection. They are trained by different algorithms. In the detectron2 model zoo, we can find models trained on COCO, Pascal VOC, LVIS dataset, etc. They are trained based on different nets, such as fast R-CNN, RPN, mask R-CNN, FPN, etc. As can be seen in Table 3.5, the model zoo contains plenty of models.

Dataset	Name	Net
COCO	faster_rcnn_R_50_DC5_3x	Faster R-CNN
	faster_rcnn_R_50_FPN_3x	
	faster_rcnn_R_101_FPN_3x	
	faster_rcnn_X_101_32x8d_FPN_3x	
	mask_rcnn_R_50_C4_1x	Mask R-CNN
	mask_rcnn_R_50_DC5_3x	
	rpn_R_50_C4_1x	RPN
	fast_rcnn_R_50_FPN_1x	
	COCO-Detection/retinanet_R_101_FPN_3x	Retina Net
	retinanet_R_50_FPN_1x	
LVIS	mask_rcnn_R_50_FPN_1x	Mask R-CNN
	mask_rcnn_X_101_32x8d_FPN_1x	
	mask_rcnn_R_101_FPN_1x	
Cityscapes	mask_rcnn_R_50_FPN	Mask R-CNN
PASCAL VOC	faster_rcnn_R_50_C4	Faster R-CNN

**Table 3.5 Models in model zoo of Detectron2**

### 3.2.4.1. Mask R-CNN

The mask R-CNN algorithm extends fast R-CNN by adding a branch that is parallel to the regression of the existing object detection frame and used to predict the target mask, and by adding a branch that is used to predict the segmentation mask on each ROI.

Mask R-CNN adopts two stages. The first stage is RPN which is the same as fast R-CNN. This step proposes the boundary box of candidate objects. The second stage is essentially the derivation of fast R-CNN. It uses ROI pool from the candidate framework to extract features and perform classification and bounding box regression. Mask R-CNN also outputs binary masks for each ROI.

By detectron2, one model trained by mask R-CNN algorithm is used to predict. The result image is shown in the figure. Similar to the traditional object detection algorithm, the model can mark the detected targets with edge boxes. In addition, it can also mark the pixels with eye-catching colors. This is the advantage of mask R-CNN algorithm.

In Figure 3.8, there are many dogs of different sizes in this picture. Their positions are different. Many dogs are blocked by others. Because the objects in this image are blocked, numerous and small, it is considered to be a very difficult picture to detect. Mask\_rcnn\_X\_101\_32x8d\_FPN\_3x is a model trained with Mask R-CNN algorithm. We use this model to detect the image. In result, the model can recognize 20 dogs with 3 mistakes and 7 omissions, it also completes semantic segmentation when the confidence is 0.4. Although the results are unqualified at the human level, the results are acceptable in the field of computer vision.

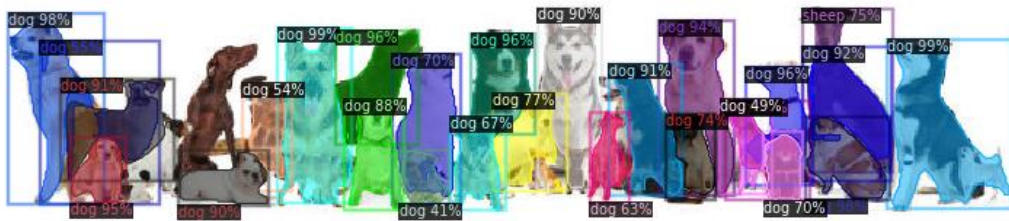


Figure 3.8 The prediction result of mask R-CNN model

### 3.2.4.2. Faster R-CNN

In detectron2, it is available to use the model trained by fast R-CNN, as shown in Figure 3.9, this is the result of detection predicted by faster\_rcnn\_X\_101\_32x8d\_FPN\_3x model.

The model trained by fast R-CNN can recognize 28 dogs with 2 omissions but no mistakes under the confidence of 0.4, which is better than other models mentioned. The result is quite acceptable.

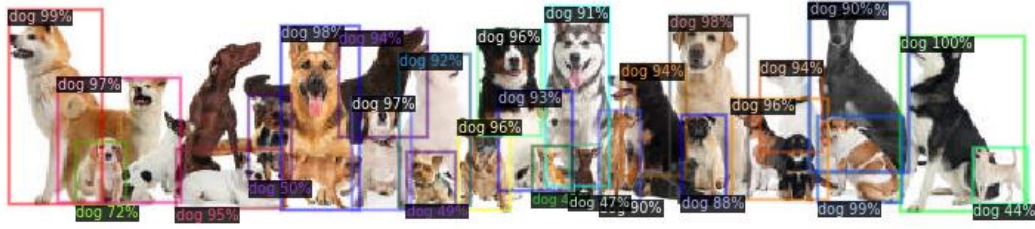


Figure 3.9 The predicting result of a faster R-CNN model

### 3.2.4.3. Retina Net

Lin proposed Focal loss as well as Retina Net in his work [11]. Focal loss is used to solve the problem that the difference between the number of positive and negative samples is too large in the process of one-stage object detection. Based on FPN, Retina Net is a new one-stage object detection framework. At present, it achieves the best balance in accuracy and speed. Retina Net tries to improve the accuracy through the focus loss algorithm, and combines the one-stage method and the two-stage method to improve the speed while ensuring the accuracy. It is essentially the one-stage method and its backbone are FPN.

Retinanet\_R\_101\_FPN\_3x is a model trained by retina net. We use it to detect the image. The results are shown in the Figure 3.10, and it can be found that the model can recognize 18 dogs with 4 mistakes and 8 omissions under 0.4 confidence, which is not as accurate as other mentioned models.

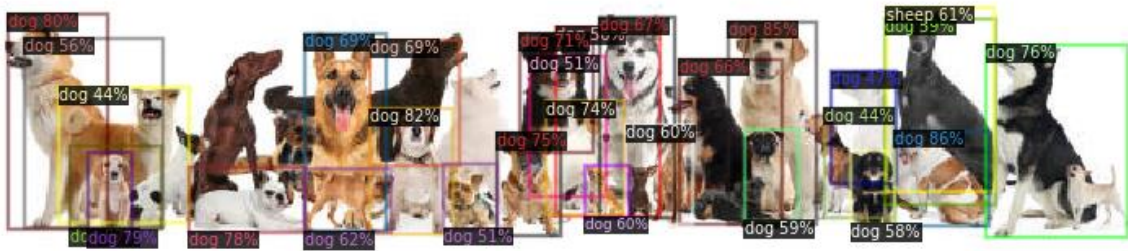


Figure 3.10 The predicting result of a Retina Net model

## 3.2.5. Comparison and application of algorithm

### 3.2.5.1. Comparison between YOLO and Detectron2

As shown in Figure 3.11, the test results of YOLOv5x6 model trained based on YOLO algorithm in this dog picture show that the model can correctly predict 17 dogs and 13 dogs are missed, its accuracy is not as good as those models involved in detectron2. Previous studies have already shown that the disadvantage of YOLO algorithm which is a one-stage method, lies



### 3.2.6. Web programming

Web app provides a visual operation interface for users, and the detection function of animal pictures is realized on the App.

#### 3.2.6.1. *Flask*



Figure 3.12 Logo of Flask

According to the developer's description, flask is a web app framework that is easy to learn, easy to use and has good scalability. Now it is the most popular Python web app framework [12].

With Flask, a server can be set up to receive the pictures uploaded by users, identify them through YOLO algorithm, and then return the marked pictures and the counting conclusion, which will be displayed on the App interface.

#### 3.2.6.2. *App architecture*

In the project, there are two templates files in HTML format, which serve as the front-end interface for user operations. The back-end consists of the code of the target recognition algorithm and the stored data and model.

As 错误!未找到引用源。 shown, there are two methods to realize the data interaction between the browser and the server, named 'log' and 'upload' respectively. The 'upload' method is to submit a form. The form includes the pictures uploaded by the user. After clicking the detection button, the image can be saved locally and the back-end detection program can be activated. Then 'render templates' method is used to return data to the Browser.

```

@app.route('/index', methods=['POST', 'GET']) # add route
def upload(): # download image then detect it
    if request.method == 'POST':
        f = request.files['file']
        download uploaded images
        detect image
        return render_template('detect.html', text=count())

    return render_template('index.html')

@app.route('/log', methods=["POST", "GET"])
def log(): # get feedback then write into text
    value = request.get_json()
    if str(value) != "None":
        write feedback to a text
    return render_template('index.html')

```

Listing 3.2 Route code

A text field containing placeholders is used to accept the data from the back end. The log method also submits requests in the form of post. To prevent conflicts between submitting two forms, AJAX (asynchronous + JavaScript + xml) is used. AJAX enables reading data from a web server - after the page has loaded, updating a web page without reloading the page and sending data to a web server - in the background [13]. This technique ensures that requests are submitted without refreshing the page, also the address of the website is not changed. In this method, the data type is set to JSON, and the value is the feedback of users.

```

$.ajax({
    type: 'post',
    url: "/log",
    data: JSON.stringify(data), //转化字符串
    contentType: 'application/json; charset=UTF-8',
})

```

Listing 3.3 AJAX code

When the user clicks the feedback button, the feedback submitted by him will be received and read through the JavaScript code in the HTML document, and the data will be sent to the server through AJAX. The server will accept the JSON document through the request method, process it and write it locally to form a record document that can be opened locally.

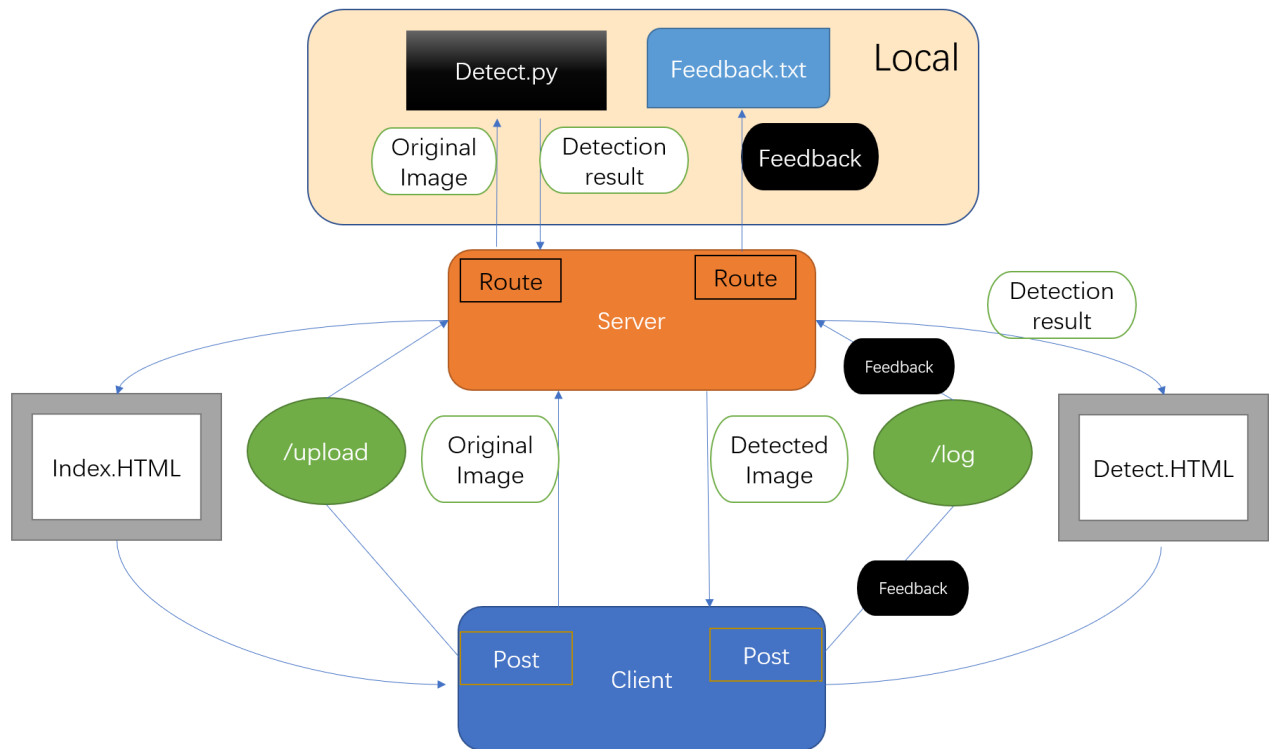


Figure 3.13 App achitecture

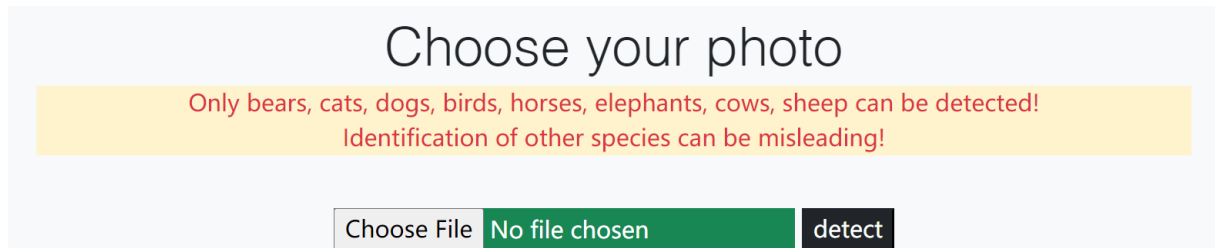
### 3.2.6.3. UI design

In UI design, bootstrap is used to provide CSS style sheets, which is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and (optionally) JavaScript-based design templates for navigation, forms, buttons, and other interface components.



Figure 3.14 Logo of Bootstrap

As shown in Figure 3.15, the app's home page contains an upload button that allows you to upload files of image type. There is a prompt box above the button to alert the user to the restrictions of the category of animals to be detected. After uploading the original image, it is allowed to directly click the detect button.



**Figure 3.15 The home page of app**

As can be seen in Figure 3.16, after the detection, the original image will be displayed below the button, and its size may be reshaped to adapt to the browser size. After detection, the image marked by the algorithm will be displayed directly below the original image, which has the same size as the original image.

At the bottom of the picture, there is a text field, which is used to contain conclusion statements, specifying the type and number of animals.




Choose your photo


Only bears, cats, dogs, birds, horses, elephants, cows, sheep can be detected!  
Identification of other species can be misleading!

Choose File No file chosen detect

Original image:



Detected image:



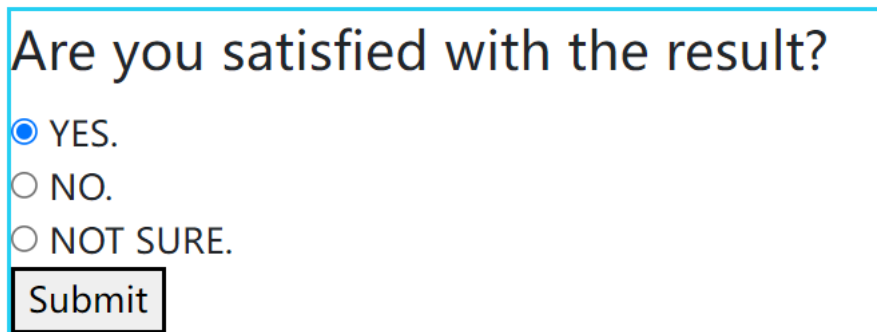
In this image, there are:

- 1 bird(s)
- 1 cat(s)

**Figure 3.16 The result page**

Figure 3.17 shows a feedback table for users to choose. It is located at the bottom of the result page. Users can feed back satisfaction information in this form. After submission, the server will receive the message log, including the submission time and submission content then

preserve the message into a text file. In this way, I can understand the real feedback and use experience of Web app users, and lay a foundation for improving the detection accuracy and algorithm. At the same time, this feedback form can improve the interactivity. After uploading the form, there will be a striking red font to express gratitude to users, and give the email contact information of developers, so as to provide channels for users to put forward improvement suggestions and improve their desire for communication.



Are you satisfied with the result?

☒ YES.

☐ NO.

☐ NOT SURE.

Submit

Figure 3.17 The feedback table at the bottom

### 3.2.7. Mobile app on Android

A mobile application is a computer program or software application designed to run on a mobile device such as a phone, tablet, or watch [14].

In this experiment, we packaged web app into a mobile app that can run on Android devices. Due to the closeness of IOS system, the mobile phone of IOS system only supports the form of web app.

Based on the bls cloud ([www.appbsl.cn](http://www.appbsl.cn)), apk files that can be downloaded by the Android operating system are packaged. Figure 3.18 shows the logo of the app. The design inspiration of this logo comes from a tested picture, in which there is a cat and a bird, and the red frame clearly marks the two animals.



AnimalDetection

Figure 3.18 Logo of mobile app on Android

As shown in Figure 3.19, it is a screenshot of using mobile app on Android phone. Image upload can be realized without obstacles, with very good adaptability, and warnings and pop-up prompts can also be displayed on Android mobile app.

In fact, when the app runs on the mobile phone, you can also select three image sources in the upload image interface: "take photos", "picture library" and "browse local files". Thus, users can easily find the picture they want to detect. Of course, taking pictures requires the user to use the camera license, and selecting pictures requires the picture browsing license.

The loading graph during loading can also be well displayed in the mobile app, and a red progress bar is designed above to display the current calculation progress. When the request process does not reach 78%, but the request time has exceeded 30 seconds, it will automatically remind the request to timeout. In some mobile browsers, it can't show the rate of progress, which is an advantage of mobile app compared with web app. Another advantage is that the display ratio of mobile app is more in line with the mobile screen, while web app is displayed according to the size of the browser when running on the mobile phone.

The mobile app also adds an overlay page at startup, which is a tiled picture with short tips. The open picture lasts for at least 1 second. During this time, send a request to the server to complete the loading. When this second is over and the start-up progress does not reach the set threshold of 60%, it will enter the second start-up time, with a total length of 10 seconds. Once the progress reaches 60%, it will automatically exit the start-up interface and enter the app operation interface.

Mobile app also has a confirmation prompt box when exiting to prevent mis-operation.

Briefly, the mobile app "animal detection" supports all devices under the Android operating system. Its compatibility is better than the web app and the user experience is better. But its disadvantage is that it needs to be downloaded and installed and is not as easy to obtain as web app, moreover it does not have such good cross platform performance

## Choose your photo


Only bears, cats, dogs, birds, horses, elephants, cows, sheep can be detected!  
Identification of other species can be misleading!

Choose File

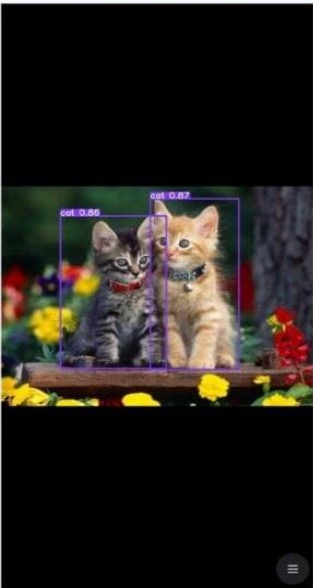
No file chosen

detect

Original image:



Detected image:



In this image, there are:

2 cat(s)

Are you satisfied with the result?

☒ YES.

☐ NO.

☐ NOT SURE.

Submit

Figure 3.19 Screenshot of mobile app on Android cellphone

29





resetting the class index value. Listing 4.1 is the simple code of the relabel method, which operates the string in the line by traversing the line, so as to reset to the corresponding value.

```
def CreateNewMaskTxt(input_filename):
    with open(input_dir_path + '/' + input_filename, 'r') as input-
file: # open input file
        with open(output_dir_path + '/' + input_filename, 'a') as
outputfile: # open output file
            for line in inputfile:
                list1 = line.rstrip('\n').split(' ')
                print(list1)
                label = list1[0]
                x1,y1,x2,y2 = coordinate value of boundary
                reset label to correct value
                list2 = [label, x1, y1, x2, y2]
                outputfile.write(label x1 x2 y1 y2)
            outputfile.close()
        inputfile.close()
    return outputfile
```

**Listing 4.1 Relabel function**

#### 4.1.1. Several parameters

True positive (TP): the real category of the sample is positive, and the result predicted by the model is also positive, and the prediction is correct.

True negative (TN): the real category of the sample is negative, and the model predicts it as negative, and the prediction is correct.

False positive (FP): the real category of the sample is negative, but the model predicts it as positive, which is wrong.

False negative (FN): the real category of the sample is positive, but the model predicts it as negative, which is wrong.

The following parameters are used to analyze the performance of the detect.

##### 1) Accuracy

Accuracy is our most common evaluation index.  $\text{Accuracy} = (TP + TN) / (P + n)$ . This is easy to understand. It is the number of samples divided by the number of all samples. Generally speaking, the higher the accuracy, the better the classifier;

## 2) Error rate

The error rate is opposite to the correct rate. It describes the proportion of misclassification by the classifier.  $\text{Error rate} = \frac{P+FN}{P+n}$ . For an instance, pair and error are mutually exclusive events, so  $\text{accuracy} = 1 - \text{error rate}$ ;

## 3) Sensitivity

$\text{Sensitive} = \frac{TP}{P}$ , which represents the proportion of pairs in all positive cases, and measures the recognition ability of the classifier to positive cases;

## 4) Specificity

$\text{Specificity} = \frac{TN}{N}$ , which indicates the proportion of pairs in all negative cases, and measures the recognition ability of the classifier to negative cases;

## 5) Precision

Precision is a measure of accuracy, which represents the proportion of positive examples in the examples divided into positive examples,  $\text{precision} = \frac{TP}{TP+FP}$

## 6) Recall

Recall rate is a measure of coverage. Multiple positive examples are divided into positive examples.  $\text{Recall} = \frac{TP}{TP+FN} = \frac{TP}{P} = \text{sensitive}$ . It can be seen that the recall rate is the same as the sensitivity.

## 7) F1-Score

F1 score is a measure of classification problems. F1 score is often used as the final evaluation method. It is the harmonic average of accuracy rate and recall rate, with a maximum of 1 and a minimum of 0.

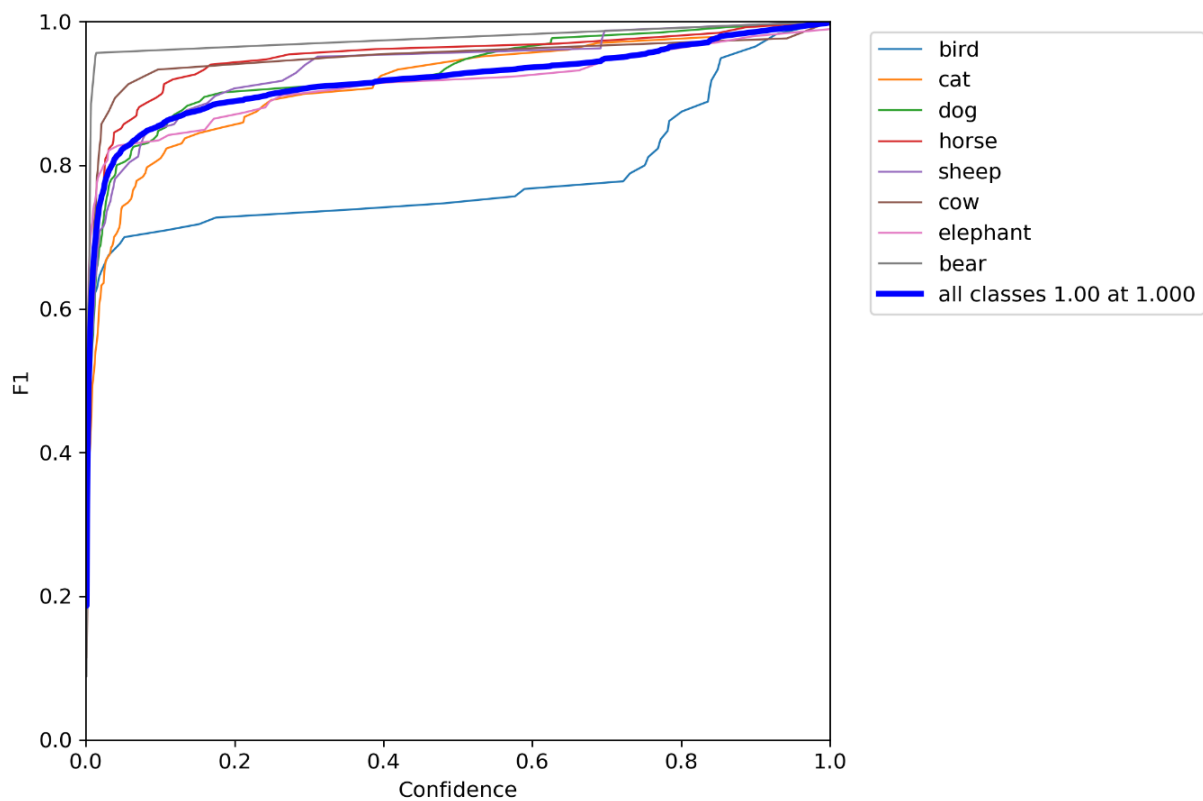
For a classification, it combines a judgment index of precision and recall. The value of F1 score is from 0 to 1, 1 is the best and 0 is the worst

F1 score takes both recall and precision into account.  $F1 \text{ score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$



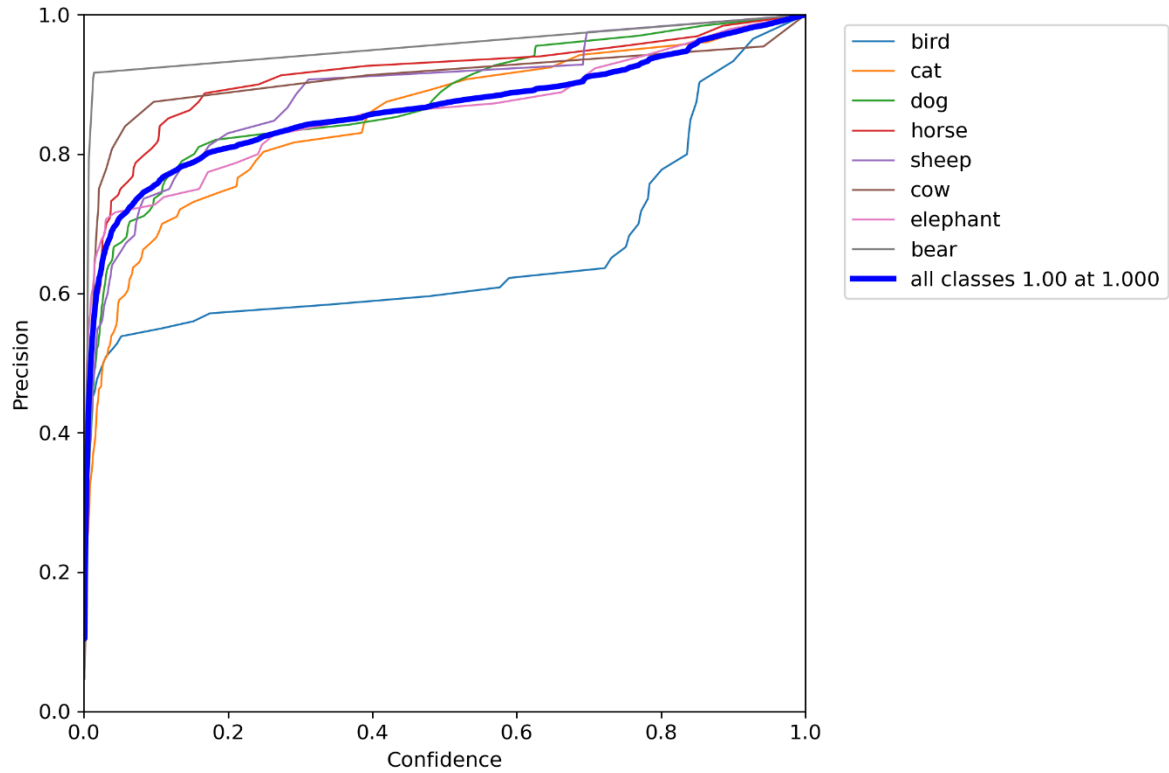


As Figure 4.4 shown, it is the F1 curve with F1 score as the y-axis and confidence as the x-axis, which reviews the relationship between the F1 score and the confidence. All classes reach the top F1 score 1.00 at confidence of 1.000. The F1 score of all animal classes increases with the increase of confidence. The F1 score cure of bears is optimal at low confidence. While the F1 score of birds is only about 0.7 when it is lower than 0.75 confidence, which is much lower than that of other animal classes. After analysis, this result can be attributed to low precision rather than low recall at low confidence. However, this value surged after that point and finally intersected with the end points of other curves.



**Figure 4.4 F1 curve**

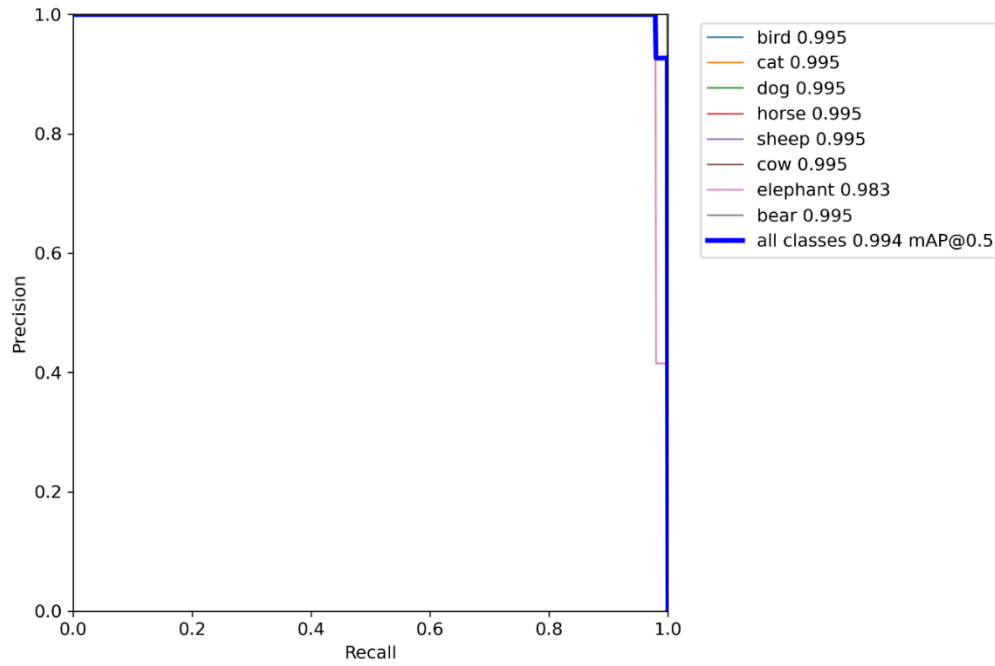
Figure 4.5 reveals the relationship between the precision and confidence. It has the precision as y-axis and confidence as x-axis. All classes reach the top precision as 1.00 at confidence of 1.000. The precision value of all animal classes increases with the increase of confidence. They all show low precision at low confidence, and vice versa. The precision value of birds is only about 0.5 when it is lower than 0.75 confidence, which is much lower than that of other animal classes. However, this value surged after that point and finally intersected with the end points of other curves.



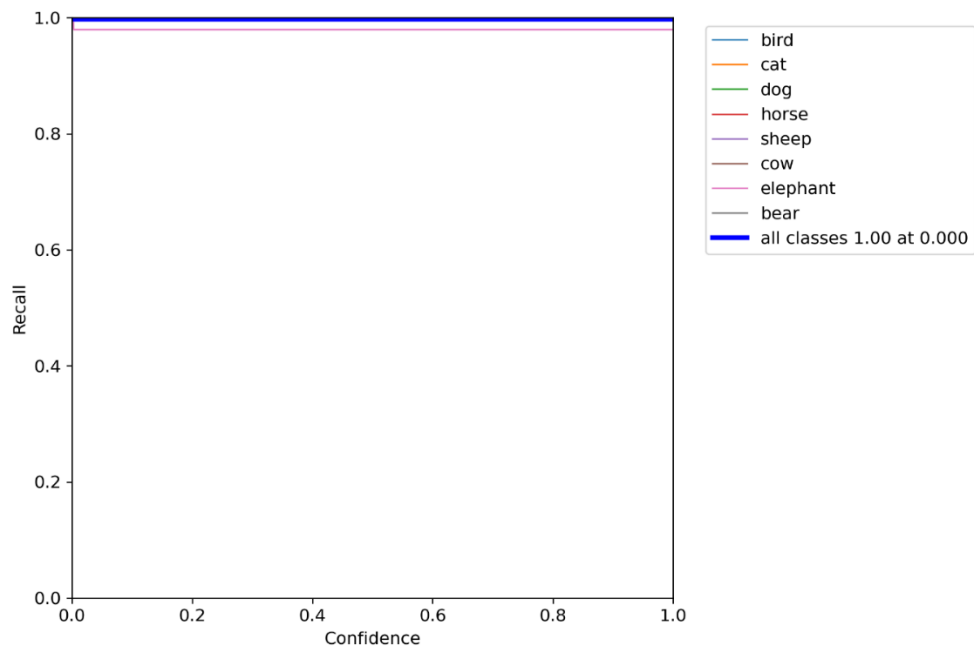
**Figure 4.5 P curve**

Figure 4.6 reveals the curvilinear relationship between precision and recall with precision as y-axis and recall as x-axis. If the P-R curve of one learner completely covers the PR curve of the other learner B, it can be asserted that the performance of A is better than that of B. But if the curve cross, it can be compared according to the area below the curve, but the more common is the equilibrium point F1. The equilibrium point (BEP) is the value when  $P = R$  (the slope is 1). The greater the F1 value, we can think that the performance of the learner is better.

In this case, the equilibrium point of this P-R curve is fairly close to 1.0 which means the model is extraordinary. Except the mAP of elephants which reaches the value of 0.983, all other animals have an average mAP of 0.995. Also, the average mAP of all classes is 0.994, which can be considered high.

**Figure 4.6 P-R curve**

Lastly, the R-Curve is shown in Figure 4.7 with recall as y-axis and confidence as x-axis. Except that the recall value of elephants is slightly lower than 1, the recall value of other animals is 1, and the overall recall value is also 1. The recall value does not change according to the confidence.

**Figure 4.7 R-curve**

## 4.2. Quality test

We conducted quality test to test the performance of the app.

### 4.2.1. Image size

According to the configuration file of YOLOv5 algorithm, all pictures will be firstly reset to 640pixels \* 640 pixels. In the neural network, the input layer is 608\*608\*3. Image size may affect detection accuracy.

#### 4.2.1.1. Width to height ratio

If the original picture is rectangular, the black frame will be added through the adaptive algorithm, which may cause data redundancy and detection error. In this test, a picture whose length is greater than the width and a picture whose width is much greater than the length are used as material to test whether the app can successfully detect.

As shown in Figure 4.8, the original image size is 300pixels \* 168 pixels and the detection result are reasonable.



Figure 4.8 Detection result of the original image

After some image clipping operations, the original image is reconstructed into a 300\*50 pixel image, which means that the aspect ratio is 6:1. As shown in the Figure 4.9, the app can still detect accurate results.



Figure 4.9 Detection result of long image with width:height = 6:1

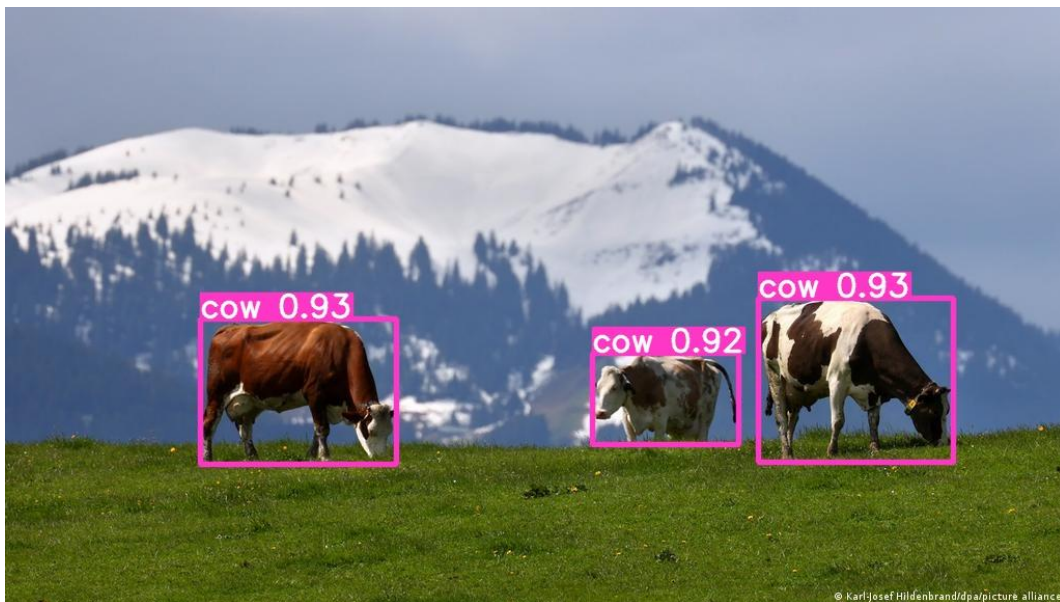
The same thing happens in a picture with a width to height ratio of 1:6.

It can be concluded that the app has the same detection accuracy for pictures with different length width ratios.

#### 4.2.1.2. Image pixels

For the same picture, with the same zoom ratio, the larger the pixel value, the clearer the human visual experience. For computer vision, the larger the pixel value means that more data need to be extracted. In this test, pictures of 12160\*6000pixels, 1024\*576pixels, 90\*50pixels, 70\*40pixels and 30\*17pixels with the same content and different pixel values were used to test the performance of the app.

As shown in Figure 4.10, the detection result of the picture with 1024\*576pixels is accurate, and the bounding box reliability of all the objects in this image has reached more than 0.9. The detection result of the picture with low pixel is also correct, but its lowest reliability is only 0.6.



**Figure 4.10 Detection result on 1024\*576 pixels image**

Pictures of 90\*50pixels and 70\*40pixels can also be detected accurately, while the detection results of 30\*17 are wrong, which means that pictures with too low pixels do not have enough characteristic values and cannot get correct detection results.

For pictures with very large pixel values, such as 12160\*6000 pixels, the app's detection results are also correct, which shows that more pixels do not reduce the detection accuracy.

This partly shows that images with high pixels have better detection results than those with low pixels.

### 4.2.2. Amount of detections

According to the YOLOv5 configuration file, up to 1000 targets can be detected in a single image. In fact, this number is almost impossible to achieve, because YOLO algorithm has low detection accuracy for a large number of targets. In this test, we try to analyze the impact of the number of targets on the test results.

First of all, the selected picture is a clear high-pixel picture with 7 dogs. The detection results of it is perfect. Then, this picture is copied into 24 copies and pasted into another picture, so that in the new picture, there are 168 dogs in total. However, as shown in Figure 4.11, only 104 dogs have been counted and the remaining 64 dogs were omitted.



Figure 4.11 Detection result on image with a large quantity of dogs

As a control experiment, we cut off the part of the dog except the top 7 on the picture of 148 dogs, and replaced the cut part with a white background. Without changing the pixel and position, the detection is performed again, and the result is correct.

This result shows that the number of objects has an impact on the performance of the app. The more the quantity of objects, the lower the accuracy, which is specifically manifested as omission.

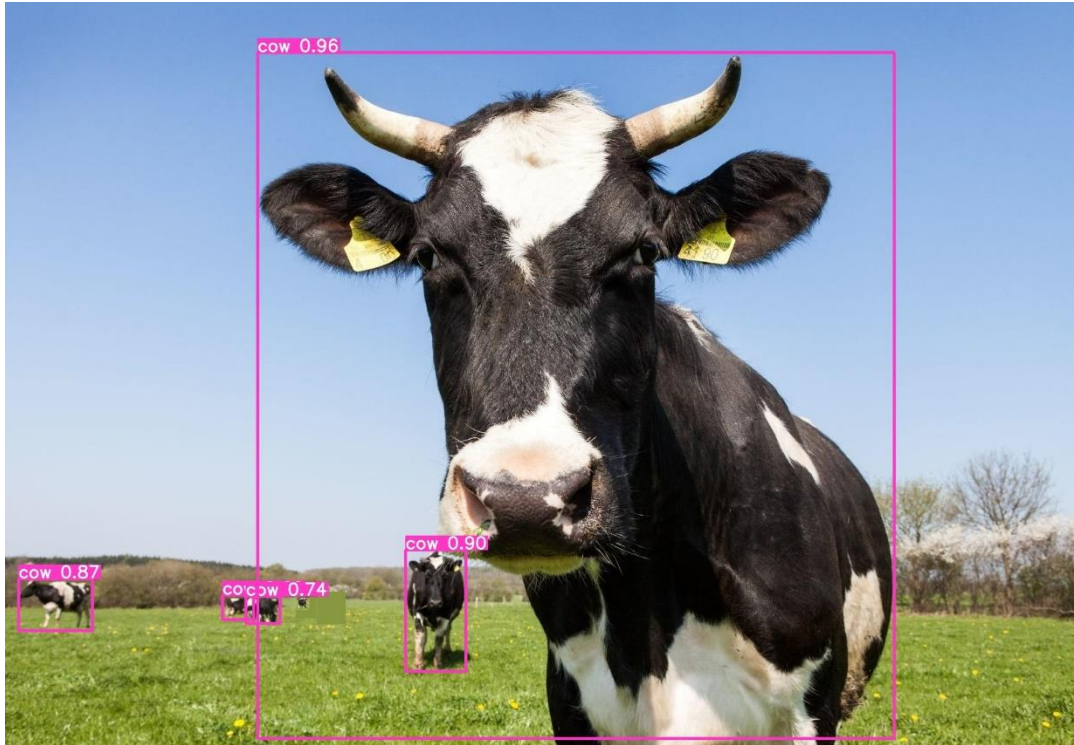
### 4.2.3. Small objects

In the field of object detection, it is generally considered that targets below  $32 \times 32$  pixels are called small objects, which is difficult to distinguish even by human eyes. Small object detection



is a major difficulty in object detection. In this test, we want to study the accuracy of app in detecting small objects.

First, we prepared a picture containing six cows of different sizes, and its test results were correct. Then we adjusted the size of one of the cows to 32\*32 pixels, and the detection result is still correct. As shown in the figure, it will not be omitted by the app until we resize one of the cows to 24\*24pixels.

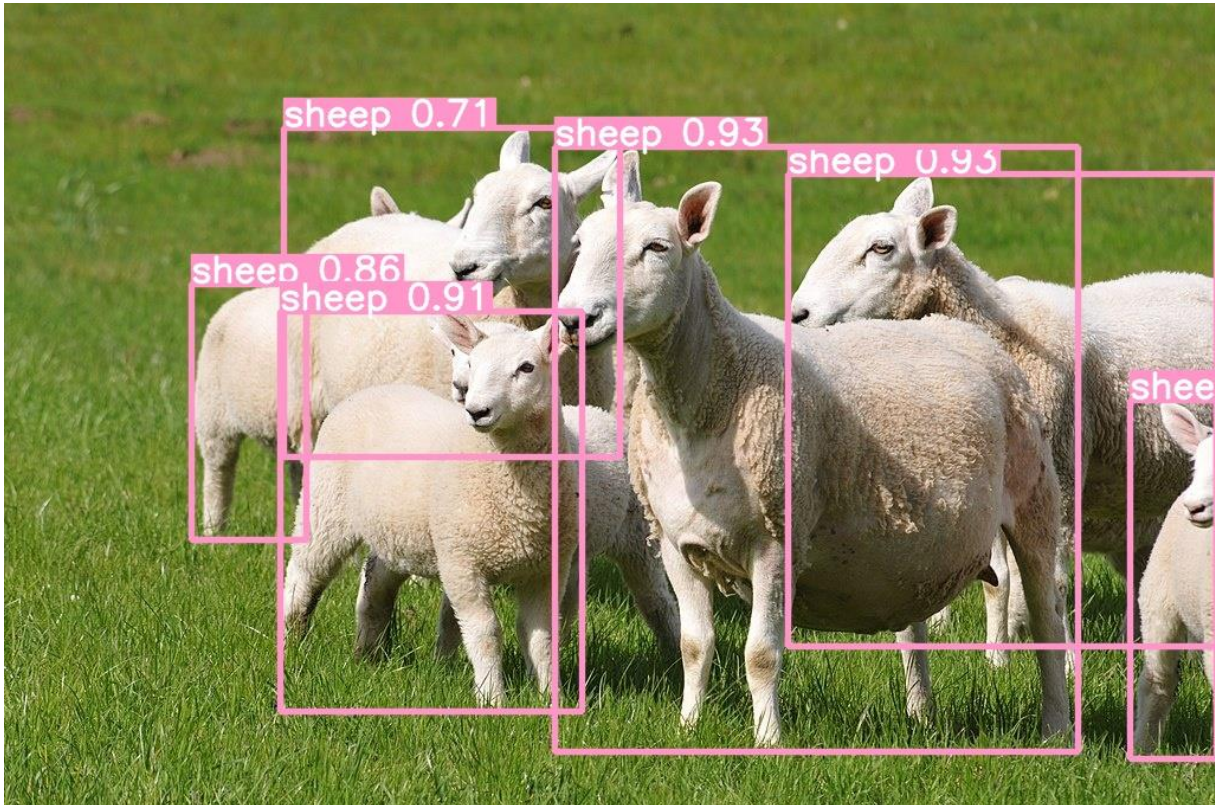


**Figure 4.12 App omit the smallest cow in this image**

This test proves that the ability of app on small object detection is acceptable, but targets smaller than 32\*32 pixels are likely to be missed in the results.

#### **4.2.4. Occlusion**

In object detection, how to detect occluded or incomplete targets is also a difficulty. We use a picture containing many sheep to test app. The test results are as shown in Figure 4.13. The incomplete sheep on the right can be detected, but the one on the left whose head is exposed with other parts been covered is missed. This sheep was thought to be one with the sheep in front of it, so it was ignored by app.



**Figure 4.13 Detection result on occluded objects**

Our conclusion is that the app has a high accuracy in detecting incomplete targets, but a low accuracy in detecting objects occluded by the same type of objects.

### **4.3. User test**

In order to find the design defects in the app, user test is conducted.

#### **4.3.1. Recruit testers**

The testers in the usability experiment included 15 college students, including 12 men and 3 women. They all have the ability to skillfully use computers and mobile phones. As shown in Figure 4.14, five testers tested with browsers on personal computers, four tested with browsers on Android phones and six tested with browsers on IOS. No tester was informed in advance of how to use this Web app.



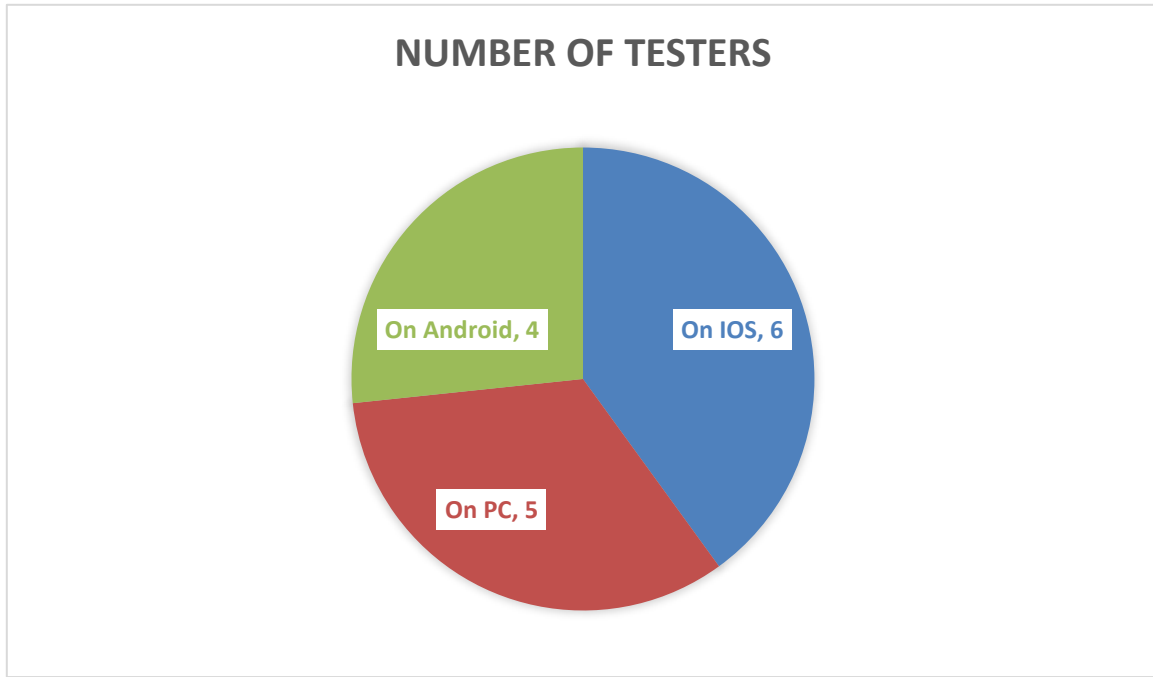


Figure 4.14 Proportion of test platform

#### 4.3.2. Scenario Design

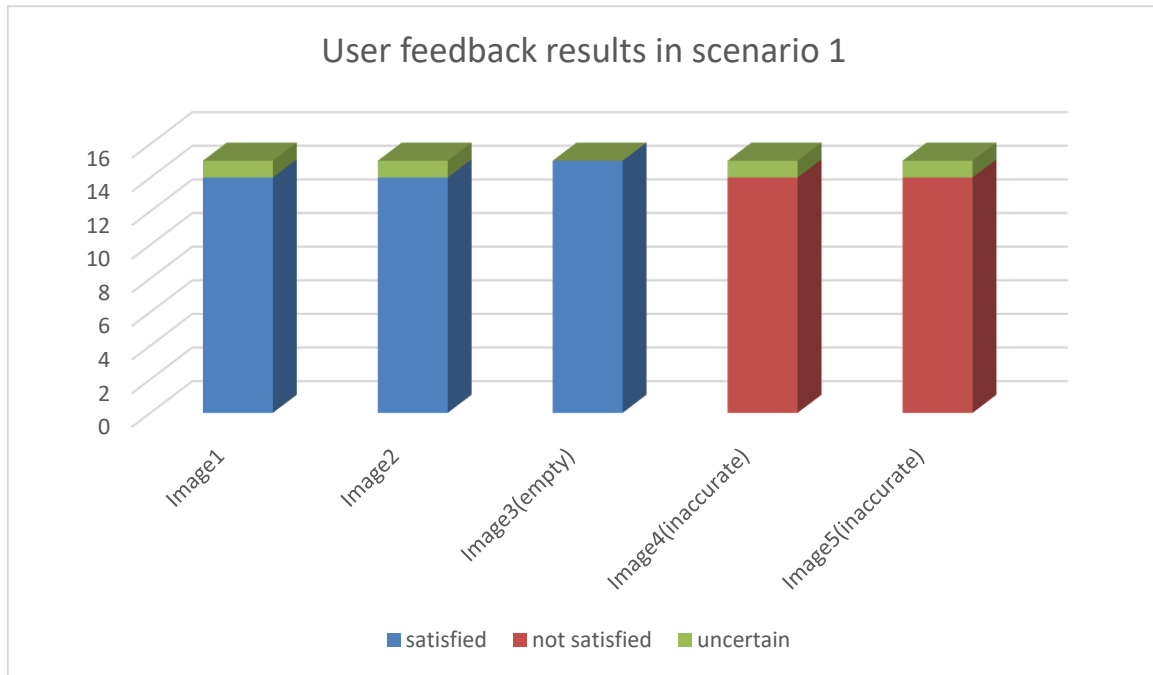
The first scenario is designed for the user to have a picture containing animals and try to learn the type and number of animals in the picture through the App. In this scenario, the user is given several pictures prepared by me in advance. Among these pictures, three can be accurately detected, including one with empty detection result and two with inaccurate detection result.

In the second scenario, users can choose their own pictures for detection. These pictures can be downloaded from the Internet, taken by themselves, etc. In this scenario, at least 5 pictures are required to be uploaded and detected, and there is no hint otherwise.

#### 4.3.3. Test Result

In the first scenario, all 15 users who were given pictures in advance could upload pictures correctly. All chose to use the feedback module below to give feedback, but no user tried to submit feedback using email address. As Figure 4.15 shown, among the feedback results, 12 users can give correct feedback on the prediction results of the picture. They are "satisfied" with the 3 correct detection results, and give "dissatisfied" or "uncertain" answers to the two wrong detection results. However, two users chose "uncertain" when they expected to choose "satisfactory" feedback. It can be seen from the results that when the App gives wrong results, users still prefer to choose uncertainty as a reply rather than trust their own judgment. Another conclusion is that users do not have great confidence in the accuracy of the model, and will

compare with their own judgment after the detection results appear. However, only in terms of usability, all users have learned to use the app without wrong operations.



**Figure 4.15 Feedback of users on detection result**

In the second scenario, 15 users uploaded a total of 53 times, and 9 of them tried to upload unsupported file formats, such as text documents, video documents, etc. After uploading, they will be prompted "uploaded the wrong format, only the image format is allowed". This test result exposed the defects of web design and did not add the prompt of uploading file type at the position of the upload file button.

After the other 44 image format files were uploaded, users uploaded a total of 40 post detection feedback, 80% were satisfied, 15% were uncertain, and 5% were dissatisfied. After analyzing these pictures, I found that some pictures cannot detect the correct answer, which is expected. Other pictures cannot get correct detection results because they contain animals not learned by the model. In fact, although there are eye-catching words on the web app to remind users that not all animals can be detected, because users do not understand the detection mechanism, they will still upload pictures including other types of animals. This can lead to serious consequences, because the model probably classifies unknown animal types into some learned animal types, resulting in wrong detection results. For example, a tiger in one picture, which is not included in the prediction labels of the model will be detected as a cat with high confidence. This problem cannot be solved at present.

#### 4.4. Improvement on App

In order to prompt users to upload image type files and prevent them from uploading other types of files, an attribute is added to the form in the HTML document `<input type="file" name="file" accept="image/*">`. In this way, the upload button only allows users to upload image type files.

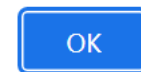
On the other hand, a small question mark is added to the right side of the detection button. As Figure 4.16 shown, when the user's mouse is overlap it, a prompt "please upload image file only!" will appear.



**Figure 4.16** The question mark next to the button

In addition, in order to prevent errors, it is because the user clicks the detect button when he does not upload any files. A pre-detection mechanism is implemented by JavaScript code. As can be seen in Figure 4.17, if no files are uploaded, the browser will give alert in advance.

Please upload a file before detection!



**Figure 4.17** Alert of none upload file found

Finally, after the user clicks the detect button, the GIF dynamic diagram of loading which can be seen in Figure 4.18 will appear on the web page, which can provide operation feedback for the user and prevent the user from having doubts while waiting.



Figure 4.18 Loading gif on the web page

## 4.5. Limitation

There are so few categories of animals that can be detected that only eight common animals are included. Most people know lots of animals, and they come into contact with many animals in their daily lives. Many animals will be photographed in the pictures, such as tigers, lions, pandas and elephants in zoos, pigs in farms and wolves in the wild. Only 8 animal types that can be detected cannot meet the needs of these image detection.

For pictures with more animals, especially for the same kind of animals, although the prediction of most animals in the picture is accurate, the counting results will be wrong due to repetition or omission. Figure 4.19 shows a Henhouse where exists a large number of chickens. Their volume is relatively small, and there are spaces far and near, and they are crowded together, which is difficult to detect. In this picture, the test result is 6 birds, but in fact, there are at least 20 birds in this picture. The target behind the chicken house in the picture, which was blocked by other birds, was not detected.



**Figure 4.19** An image of Henhouse

In addition, the detection accuracy of small and repeated targets is not high. Therefore, if this app is applied to animal husbandry to detect a large number of livestock in the same picture, the result is not reasonable.

## 5. Conclusion and outlook

The goal of this experiment is to complete a mobile app for detecting animals in images based on object detection. In this section, we will confirm the achievement of the goal. At the same time, there are still a lot of work that may be tried in the future, some assumptions are shown in this part.

### 5.1. Conclusion

#### 5.1.1. Contributions

In this experiment, I designed and developed the user interface of the app, and completed the construction of a simple server based on the flask framework. I implement a web app and an Android mobile app. I studied the related work of object detection, compared their advantages and disadvantages and chose YOLO algorithm. I annotated the validation data set of animal images and used it to test the performance of YOLO algorithm on image sets of different animals. I analyzed these performance indicators, compared the accuracy under different IOU algorithms and the acceptability under different confidence thresholds, and selected more appropriate parameters. Finally, I applied YOLO algorithm to mobile app, implemented it on the server, conduct user tests on the mobile phone and PC, analyzed the feedback, improved the app, and proposed the development direction.

#### 5.1.2. Key findings

We introduce a web app as well as a mobile app on Android that can identify the category and number of animals in the picture. The detection time of a single picture is basically less than 5 seconds. On the validation set of manual annotation created by myself mAP@.5:95 reaches 0.7 and mAP@.5 reaches 0.935.

### 5.2. Outlook

In order to improve the accuracy, we can combine YOLO with faster R-CNN algorithm, but this may increase the amount of calculation and lead to a significant increase in calculation time. In fact, this assumption is feasible on the premise that there is a powerful GPU (8-core). Fast R-CNN can achieve 5fps in GPU operation, which means that detection can be completed within 0.2 seconds. Adding the coupling time with YOLO algorithm, the whole operation will not be too long in this case. Therefore, it can be realized on Mobile apps.

Another way to improve the accuracy is to create a new animal image dataset and retrain the model. According to YOLOv5 the author's recommendation, the training set needs at least 1000

images and at least 15000 labels. This method needs a lot of training time to get a good model, so it can be applied only after the hardware computing power is improved.

After passing the IOS audit, the IOS version of mobile app can be released.

## Acknowledgments

This section contains acknowledgements from the main authors of this document.

### **Acknowledgments by Hao Yuan**

I want to thank Prof. Dr. Menno Heeren, who is my main supervisor of this thesis. He answered many of my questions and gave me many helpful suggestions.

I want to thank Google pictures for providing some data sources, GitHub for providing compilation help, and Google collab for providing me with GPU for model verification.

I would like to thank roboflow for providing a free annotated image web app and pychrome for providing a free professional IDE.

I would like to thank Dr. Jeseoph Redmon, the founder of YOLO algorithm, for his outstanding contribution to CV discipline.



## Appendix A - List of figures

Figure 3.1 Example of a original image – dog image .....	6
Figure 3.2 Logic diagram for automatically downloading web page pictures .....	7
Figure 3.3 Annotating Image by roboflow .....	9
Figure 3.4 Thumbnails of some tagged animal pictures .....	9
Figure 3.5 The comparison of different versions of YOLOv5 model.....	15
Figure 3.6 The result of detection on a bird image by model .....	16
Figure 3.7 Schematic diagram of IOU algorithm.....	17
Figure 3.8 The prediction result of mask R-CNN model .....	19
Figure 3.9 The predicting result of a faster R-CNN model.....	20
Figure 3.10 The predicting result of a Retina Net model .....	20
Figure 3.11 Detection result of YOLOv5x6 model.....	21
Figure 3.12 Logo of Flask .....	22
Figure 3.13 App achitecture .....	24
Figure 3.14 Logo of Bootstrap .....	24
Figure 3.15 The home page of app.....	25
Figure 3.16 The result page.....	26
Figure 3.17 The feedback table at the bottom .....	27
Figure 3.18 Logo of mobile app on Android.....	27
Figure 3.19 Screenshot of mobile app on Android cellphone .....	29
Figure 4.1 Image detected and marked by the model.....	30
Figure 4.2 Overview of validation results .....	33
Figure 4.3 Images after validation.....	33
Figure 4.4 F1 curve .....	34
Figure 4.5 P curve .....	35

Figure 4.6 P-R curve .....	36
Figure 4.7 R-curve.....	36
Figure 4.8 Detection result of the original image.....	37
Figure 4.9 Detection result of long image with width:height = 6:1 .....	37
Figure 4.10 Detection result on 1024*576 pixels image.....	38
Figure 4.11 Detection result on image with a large quantity of dogs .....	39
Figure 4.12 App omit the smallest cow in this image .....	40
Figure 4.13 Detection result on occluded objects .....	41
Figure 4.14 Proportion of test platform.....	42
Figure 4.15 Feedback of users on detection result .....	43
Figure 4.16 The question mark next to the button .....	44
Figure 4.17 Alert of none upload file found.....	44
Figure 4.18 Loading gif on the web page.....	45
Figure 4.19 An image of Henhouse.....	46

## Appendix B - List of listings


Listing 3.1 Get images from Google by selenium.....	8
Listing 3.2 Route code.....	23
Listing 3.3 AJAX code .....	23
Listing 4.1 Relabel function .....	31

## Appendix C - List of tables

Table 3.1 System parameter .....	4
Table 3.2 Package requirements.....	5
Table 3.3 Hardware parameter .....	5

Table 3.4 Labels of animals in the train dataset .....	17
Table 3.5 Models in model zoo of Detectron2 .....	18

## Bibliography

- [1]B. Thomee and M. S. Lew, “Interactive search in image retrieval: a survey,” *Int J Multimed Info Retr*, vol. 1, no. 2, pp. 71–86, Jul. 2012, doi: 10.1007/s13735-012-0014-4.
- [2]P. Rajeshwari, P. Abhishek, P. S. | T. Vinod, and Anurag Group of Institutions, Telangana, India, “Object Detection: An Overview,” *IJTSRD*, vol. Volume-3, no. Issue-3, pp. 1663–1665, Apr. 2019, doi: 10.31142/ijtsrd23422.
- [3]S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in *Advances in Neural Information Processing Systems*, 2015, vol. 28. Accessed: May 12, 2022. [Online]. Available: <https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html>
- [4]Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object Detection in 20 Years: A Survey,” *arXiv:1905.05055 [cs]*, May 2019, Accessed: May 12, 2022. [Online]. Available: <http://arxiv.org/abs/1905.05055>
- [5]G. van Rossum and F. L. Drake, *An introduction to Python: release 2.5*, 2. print. Bristol: Network Theory Limited, 2006.
- [6]“Computer vision,” *Wikipedia*. Apr. 09, 2022. Accessed: May 14, 2022. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Computer\\_vision&oldid=1081781547](https://en.wikipedia.org/w/index.php?title=Computer_vision&oldid=1081781547)
- [7]J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016, pp. 779–788. Accessed: May 08, 2022. [Online]. Available: [https://www.cv-foundation.org/openaccess/content\\_cvpr\\_2016/html/Redmon\\_You\\_Only\\_Look\\_CVPR\\_2016\\_paper.html](https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html)
- [8]“ultralytics/YOLOv5: YOLOv5  in PyTorch > ONNX > CoreML > TFLite.” <https://github.com/ultralytics/YOLOv5> (accessed May 19, 2022).
- [9]“What is the COCO Dataset? What you need to know in 2022,” *viso.ai*, Jul. 29, 2021. <https://viso.ai/computer-vision/coco-dataset/> (accessed May 12, 2022).
- [10] *facebookresearch/detectron2*. Meta Research, 2022. Accessed: May 31, 2022. [Online]. Available: <https://github.com/facebookresearch/detectron2>
- [11]T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” *arXiv*, *arXiv:1708.02002*, Feb. 2018. doi: 10.48550/arXiv.1708.02002.

[12]“Release 2.1.1 · pallets/flask,” GitHub. <https://github.com/pallets/flask/releases/tag/2.1.1> (accessed May 09, 2022).

[13]“AJAX Introduction.” [https://www.w3schools.com/js/js\\_ajax\\_intro.asp](https://www.w3schools.com/js/js_ajax_intro.asp) (accessed May 30, 2022).

[14]“Mobile app,” Wikipedia. May 11, 2022. Accessed: May 19, 2022. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Mobile\\_app&oldid=1087277547](https://en.wikipedia.org/w/index.php?title=Mobile_app&oldid=1087277547)