



Département Informatique  
64 avenue Jean Portalis  
37200 Tours, France  
Tél. +33 (0)2 47 36 14 14  
[polytech.univ-tours.fr](http://polytech.univ-tours.fr)

**ORA project \_ IT-English project**

**2019-2020**

# **Research and implementation of automatic image labeling algorithms**

**Academic tutors**  
**JEAN-YVES RAMEL**  
**TIFAINÉ BACHET**

**Student**  
**GUO XIAOQING (DI4)**

19 May 2020



# Table of contents

<b>TABLE OF CONTENTS .....</b>	<b>1</b>
<b>INTRODUCTION.....</b>	<b>1</b>
<b>1. STATE OF THE ART .....</b>	<b>2</b>
<b>2. METHODOLOGY.....</b>	<b>4</b>
2.1 ENVIRONMENT SETUP .....	4
2.2 ALGORITHM RESEARCH.....	7
<b>3. RESULTS.....</b>	<b>14</b>
3.1 ORIGINAL PROJECT OPERATION AND RESULT DISPLAY .....	14
3.2 NEW PROJECT OPERATION AND RESULT DISPLAY .....	17
<b>4. FOLLOW-UP .....</b>	<b>21</b>
<b>CONCLUSION.....</b>	<b>23</b>
<b>BIBLIOGRAPHY .....</b>	<b>24</b>
<b>ABSTRACT.....</b>	<b>25</b>

# Introduction

**Project title:** Research and implementation of automatic image labeling algorithms

**Prepared by:** JEAN-YVES RAMEL , TIFAINÉ BACHET , GUO XIAOQING

## Summary:

- **Background:**

*< TP3 / Question 6 / Image Tagging >*

It's up to you to create a plugin associating keywords describing the content ( *color, content, quality...* ) of the images and showing your image processing skills. This plugin will be described in detail in your report.

Learning base available on *Celene*. This plugins will generate ".txt" files in a format similar to those of the learning base.

*An image = a.txt file with the same name as the image ; only the extension changes ; the txt file is created in the same folder as the image file.*

- **New project description and goal:**

The project is a continuation of exercise 6 of TP3 image analysis ( *DI4 - see on Celene* ).

The objective of the project is to study and implement ( *in python + opencv* ) a software prototype associating keywords with images or photos.

The keywords must describe the content ( *color, content, quality, etc.* ) of the images.

This prototype should generate ".txt" files in a format similar to that of the learning base available on *Celene*.

*An image = a.txt file with the same name as the image; only the extension changes; the txt file is created in the same folder as the image file.*

The prototype produced will be described in detail in your report.

Info and Learning base available on *Celene* ( <https://celene.univ-tours.fr/course/view.php?id=4926> ).



# 1. State of the art

In the last ten years, artificial intelligence, machine learning, and data science are strongly improving most technologies in the whole world. The generation and integration of datasets, the hardware innovation, the large adoptions and improvement of research into neural architecture and so on greatly promote development of them. This allows computers to get mind abilities in everything from vision to natural language processing to audio understanding to complex signal processing.

Last year we can see the growth in AI research, development and deployment, especially for image recognition, image generation, natural language understanding, etc. At the same time, there are many challenges in data management, efficiency measurement, computing power and other issues.

In addition, because related researches have become so widespread, so it's more important today to popularize the application of related algorithms in more and more domains.

The following are the main development trends in main domains of AI and machine learning in 2020 and beyond:

- **Domain [ NLP ( Natural Language Processing ) ]**

Natural language processing (NLP) is concerned with the interactions between computers and human/natural languages, especially with programming computers to process and analyzing large amounts of natural language data.

Challenges in natural language processing frequently involve speech recognition, natural language understanding, and natural language generation.

**Current research trends are as follows:**

- The new NLP paradigm is “pre-training + fine-tuning”.
- Linguistics and knowledge are likely to advance the performance of NLP models.
- Neural machine translation is demonstrating visible progress.

- **Domain [ Conversational AI ]**

Conversational AI is a set of technologies behind automatic messaging and voice-enabled applications and it provides human-like interactions between computers and humans. It can communicate like human beings by recognizing speech and text, understanding

intent, decrypting different languages, and responding.

**Current research trends are as follows:**

- Dialog systems are improving at tracking long-term aspects of a conversation.
- Many research teams are addressing the diversity of machine-generated responses.
- Emotion recognition is seen as an important feature for open-domain chatbots.

● **Domain [ CV ( Computer Vision ) ]**

Computer vision involves how to get a high-level understanding of a computer from digital images or videos, and it is dedicated to automating tasks that the human vision system can do.

Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, VS, 3D scene modeling, image restoration and so on.

During the last few years, computer vision (CV) have improved the healthcare, security, transportation, retail, banking, agriculture, and more in many ways.

**Current research trends are as follows:**

- 3D is currently one of the leading research areas in CV.
- The popularity of unsupervised learning methods is growing.
- Computer vision research is being successfully combined with NLP.

● **Domain [ RL ( Reinforcement Learning ) ]**

Reinforcement learning (RL) is still less valuable for business applications than supervised learning or even unsupervised learning. It is successfully applied only in areas where large amounts of simulation data can be generated.

**Current research trends are as follows:**

- Multi-agent reinforcement learning (MARL) is rapidly advancing.
- Off-policy evaluation and off-policy learning are recognized as very important for future RL applications.
- Exploration is an area where serious progress can be achieved.

This research project involves the field of computer vision and mainly expects to implement image attribute analysis through deep learning.

## 2. Methodology

### 2.1 Environment setup

#### 1) System Configuration:

- System: *Windows 10*
- Graphics driver: *NVIDIA GeForce 930MX, Inter® HD Graphics 620*
- Processor: *Inter® Core™ i5-7200U CPU @ 2.50GHz 2.71 GHz*

#### 2) OpenCV ( *OpenCV-Python – cv2* )

*OpenCV* is an open source library for image processing, analysis, and machine vision.

- In win10 console, install *OpenCV*: *>pip install opencv-python.*
- Test if the environment of *OpenCV-Python* is set up successfully.



The image is successfully displayed, so the environment is successfully set up.

#### ● Applications

*OpenCV's* application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human-computer interaction (*HCI*)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (*SFM*)
- Motion tracking
- Augmented reality

To support some of the above areas, *OpenCV* includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (*SVM*)
- Deep neural networks (*DNN*)

### 3) Tensorflow

My tensorflow version: tensorflow 2.1.0

- In win10 console, install tensorflow: *>pip install tensorflow.*
- Test if the installation is set up successfully.

```
F:\Python\python\1\Scripts>python -c "import tensorflow as tf; print(tf.__version__)"
2020-03-09 08:11:34.967203: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2020-03-09 08:11:34.977122: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2.1.0
```

*TensorFlow* is an end-to-end platform that makes it easy to build and deploy *ML* models.

An entire system can help solve challenging, real-world problems with machine learning.

- **Easy model building**

*TensorFlow* offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level *Keras API*, which makes getting started with *TensorFlow* and machine learning easy.

If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large *ML* training tasks, use the *Distribution Strategy API* for distributed training on different hardware configurations without changing the model definition.

- **Robust *ML* production anywhere**

*TensorFlow* has always provided a direct path to production. Whether it's on servers, edge devices, or the web, *TensorFlow* lets you train and deploy your model easily, no matter what language or platform you use.

Use *TensorFlow Extended (TFX)* if you need a full production *ML* pipeline. For running inference on mobile and edge devices, use *TensorFlow Lite*. Train and deploy models in JavaScript environments using *TensorFlow.js*.

- **Powerful experimentation for research**

Build and train state-of-the-art models without sacrificing speed or performance. *TensorFlow* gives you the flexibility and control with features like the *Keras Functional API*

and *Model Subclassing API* for creation of complex topologies. For easy prototyping and fast debugging, use eager execution.

*TensorFlow* also supports an ecosystem of powerful add-on libraries and models to experiment with, including *Ragged Tensors*, *TensorFlow Probability*, *Tensor2Tensor* and *BERT*.

- **Models & datasets**

Explore repositories and other resources to find available models, modules and datasets created by the *TensorFlow community*. We can directly call the existing pre-trained model or *API*, or download the datasets for model training.

#### 4) Python [ *Deep learning research* ]

My python version: Python 3.7.1

(v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15) [MSC v.1915 64 bit (AMD64)] on win32

```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15) [MSC v.1915 64 bit (AMD64)] on win32
```

Develop and evaluate *deep learning models* in *Python*.

The platform for getting started in applied deep learning is *Python*.

*Python* is a fully featured general purpose programming language, unlike *R* and *Matlab*. It is also quick and easy to write and understand, unlike *C++* and *Java*.

The *SciPy* stack in Python is a mature and quickly expanding platform for scientific and numerical computing. The platform hosts libraries such as *scikit-learn* the general purpose machine learning library that can be used with your deep learning models.

It is because of these benefits of the *Python* ecosystem that two top numerical libraries for deep learning were developed for *Python*, *Theano* and the newer *TensorFlow* library released by *Google* (and adopted recently by the *Google DeepMind research group*).

*Theano* and *TensorFlow* are two top numerical libraries for developing deep learning models, but are too technical and complex for the average practitioner. They are intended more for research and development teams and academics interested in developing wholly new deep learning algorithms.

The saving grace is the *Keras* library for deep learning, that is written in pure *Python*, wraps and provides a consistent agnostic interface to *Theano* and *TensorFlow* and is aimed at machine learning practitioners that are interested in creating and evaluating deep learning models.

It is a little over one year old and is clearly the best-of-breed library for getting started with deep learning because of both the speed at which you can develop models and the numerical power it is built upon.

#### 5) Other necessary packages

***collections*** : This module implements specialized container datatypes providing alternatives to Python's general purpose built-in containers, *dict*, *list*, *set*, and *tuple*.

***NumPy*** : *NumPy* adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.



**torch** : It provides a flexible N-dimensional array or *Tensor*, which supports basic routines for indexing, slicing, transposing, type-casting, resizing, sharing storage and cloning.

**torchvision.transforms** : *Transforms* are common image transformations

**os** : The *OS* module in *Python* provides a way of using operating system dependent functionality. The functions that the *OS* module provides allows you to interface with the underlying operating system that Python is running on – be that *Windows*, *Mac* or *Linux*.

**PIL** : The *Python Imaging Library (PIL)* adds image processing capabilities to your *Python* interpreter. This library supports many file formats, and provides powerful image processing and graphics capabilities.

**urllib.request** : *urllib.request* is a *Python* module for fetching *URLs (Uniform Resource Locators)*. It offers a very simple interface, in the form of the *urlopen* function.

**random** : This module implements *pseudo-random* number generators for various distributions.

**tarfile** : The *tarfile* module makes it possible to read and write tar archives, including those using *gzip*, *bz2* and *lzma* compression.

**sys** : System-specific parameters and functions. This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

**matplotlib.pyplot** : *Matplotlib* is a comprehensive library for creating static, animated, and interactive visualizations in *Python*.

**tkinter** : The *tkinter* package is a thin object-oriented layer on top of *Tcl/Tk*. *tkinter* is a set of wrappers that implement the *Tk* widgets as *Python* classes.

## 2.2 Algorithm research

### 2.2.1 Main colors recognition

#### Method:

- a) Convert the image color mode from *RGB* to *HSV*
- b) Use *cv2.inRange()* function for background color filtering
- c) Binarize the filtered colors
- d) Use *cv2.dilate()* to make dilation and erosion
- e) Statistics white area

#### Explanation:

Currently there are many types of color spaces in the field of computer vision. *HSL* and *HSV* are the two most common color models represented by cylindrical coordinates. It remaps the *RGB* model so that it can be more visually intuitive than the *RGB* model. Therefore, the effective processing of the image in the color space is generally

performed in the *HSV* space, and then a strict range needs to be given for the corresponding *HSV* component in the basic color. The following is the blur range calculated by experiment:

	black	gray	white	red		orange	yellow	green	cyan	blue	purple
<b>hmin</b>	0	0	0	0	156	11	26	35	78	100	125
<b>hmax</b>	180	180	180	10	180	25	34	77	99	124	155
<b>smin</b>	0	0	0	43		43	43	43	43	43	43
<b>smax</b>	255	43	30	255		255	255	255	255	255	255
<b>vmin</b>	0	46	221	46		46	46	46	46	46	46
<b>vmax</b>	46	220	255	255		255	255	255	255	255	255

Note: H: 0 - 180, S: 0 - 255, V: 0 - 255

## 2.2.2 Scene recognition ( type of environment and special scene )

### Method:

- Call the pre-trained model *PlacesCNN* trained on dataset *Places365*.
- Label of scene attributes is generated from dataset *SUNattribute*.
- *Heatmap* is generated using the *CAM technique*.

### Explanation:

#### ● Dataset and Model:

In the field of the computer vision, object detection and scene recognition are two much important parts, and if we expects machine learning algorithms to reach near-human semantic classification performance, we need more and more valid data to help it improve. So we pursue large datasets.

*Places365 Database* is a repository of 10 million scene photographs which is labeled with scene semantic categories, comprising a large and diverse list of the types of environments encountered in the world.

And *PlacesCNN* is a pre-trained model trained on dataset *Places365*, which uses the state-of-the-art *Convolutional Neural Networks (CNNs)*, and can generates a *heatmap* using the *CAM* technique, which is a visualization of the *CNNs* trained on *Places365*.



Fig. 1. Image samples from various categories of the Places Database (two samples per category). The dataset contains three macro-classes: Indoor, Nature, and Urban.

(pic. is coming from the website pdf file [Places](#).)

## 2.2.3 Main objects detection

**Method:** (Different steps are coming from the [learn OpenCV](#) )

- a) Download the models and try to call the pre-trained model trained on the dataset *MSCOCO* using *Mask-RCNN* and *Tensorflow*.
- b) Initialize the parameters.  
(The *Mask-RCNN* algorithm produces the predicted detection outputs as the bounding boxes. We will set a confidence score in the beginning and each bounding box will be associated with it. Only the boxes above the confidence threshold parameter will be used for further processing. )
- c) "Load the model and classes. "  
The file *mscoco\_labels.names* contains all the objects (classes) which the mode can process.  
The *colors.txt* file contains all the colors we set for masking objects in the beginning.  
Load the network:
  - [1] *frozen\_inference\_graph.pb*: Use the frozen graph file to get the pre-trained model weights.
  - [2] *mask\_rcnn\_inception\_v2\_coco\_2018\_01\_28.pbtxt*: The text graph file that has been tuned by the *OpenCV's DNN* support group, so that the network can be loaded using *OpenCV*.
- d) Read the input for the image being processed.  
( In this step we read the image, video stream or the webcam. In addition, we also open the video writer to save the frames with detected output bounding boxes. )
- e) Process each frame  
(The input image to a neural network needs to be in a certain format called a *blob*.  
So we read a frame from the input image or video stream (a video stream is

composed of many frames) and convert it to an input *blob* for the neural network using *blobFromImage* function. The video stream can be an video file or a camera monitor updated in real time.

The blob is then passed into the network as its input and to get a list of predicted bounding boxes and the object masks from the output layers. )

- f) Post-processing the network's output → Delete those boxes whose confidence score is less than the given threshold.
- g) Draw the predicted boxes  
(Finally, we draw the boxes and set their detected object labels and confidence scores on the input frame. We also draw the colored masks in the boxes. )

### Explanation:

- **Instance segmentation:**

In instance segmentation the goal is to detect specific objects in an image and create a mask around the object of interest.

Instance segmentation can also be thought as object detection where the output is a mask instead of just a bounding box.

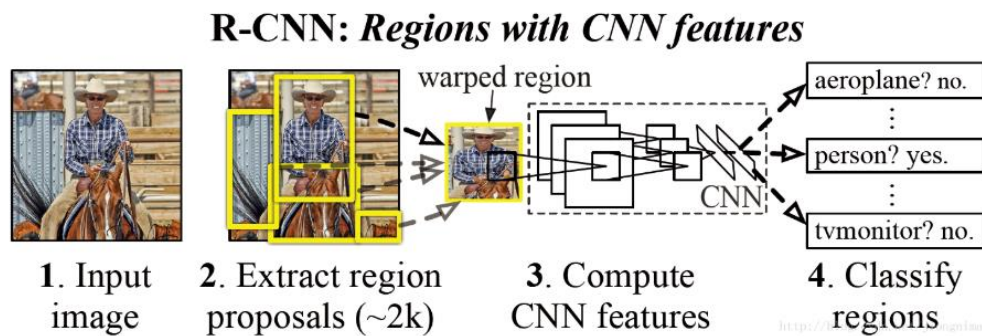
- **Regional proposal based:**

Selection of target detection area: detect the color space and similarity matrix, then detect the area to be detected based on these, and then classify and predict based on the detection result.

- **How Mask-RCNN works ?**

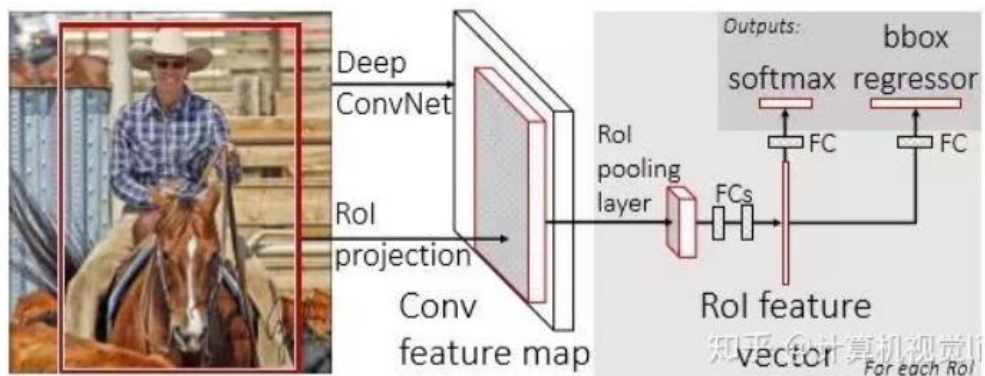
Development sequence:

1. *R-CNN* (Long time-consuming, low efficiency) --> Target detection only



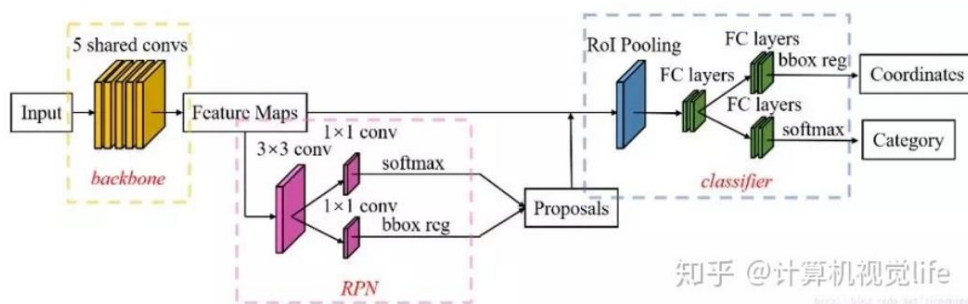
*R-CNN* generates region proposals based on selective search and then processes each proposed region, and then at one time for one image, uses Convolutional Networks to output an object label and its bounding box.

2. *Fast R-CNN* [2015] --> Target detection only



(pic. is coming from the blog website [Mask R-CNN](#))

### 3. *Faster R-CNN* [2016] --> target detection only



(pic. is coming from the blog website [Mask R-CNN](#))

#### Process:

- ① First use *RPN* to extract the candidate box region
- ② Then directly use a neural network to perform feature extraction on the entire image;
- ③ Then use a *RoI Pooling Layer* to obtain the features corresponding to each *RoI* on the feature map of the full picture,
- ④ Then use *FC* to classify and correct the bounding box.

### 4. *Mask R-CNN* [2017] --> target detection + target classification + pixel-level segmentation

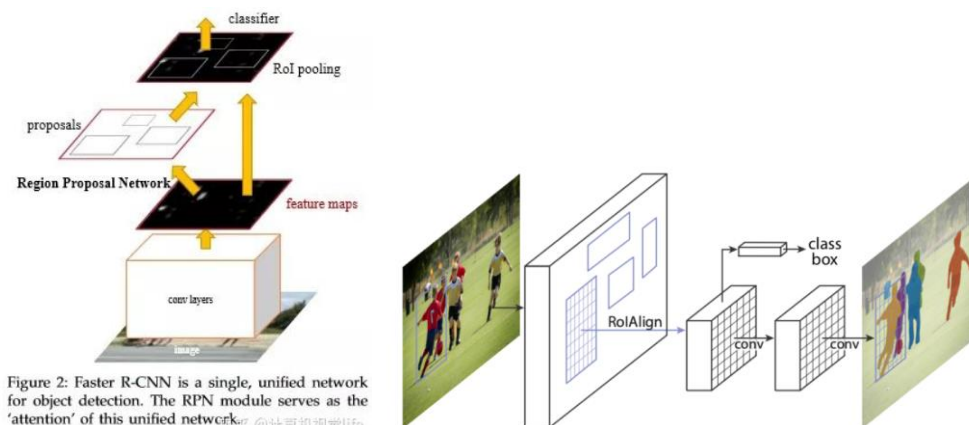


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

(pic. is coming from the blog website [Mask R-CNN](#))



### Improve:

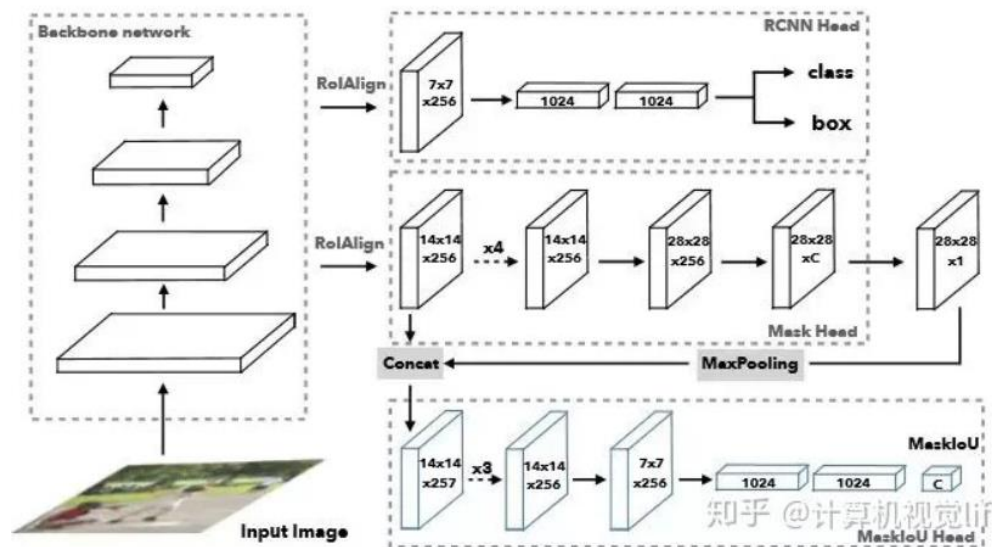
① Introduce the *Mask prediction branch (Mask-Head)* to predict the segmentation mask in a pixel-to-pixel manner.

② *ROI Pooling* is replaced by *ROI Align*, and the coarse quantization of *ROI Pooling* is removed, so that the extracted features are well aligned with the input;

In the instance segmentation on Mask R-CNN framework, three main things need to be done:

- 1) Target detection: the bounding boxes are drawn directly on the result image.
- 2) Target classification: for each target, to find the corresponding class/label to distinguish whether it is a person, a car, or other categories.
- 3) Pixel-level target segmentation: for each target, to distinguish at the pixel level, what is the foreground and what is the background.

5. *Mask Scoring R-CNN* [2019] --> target detection + target classification + pixel-level segmentation



(pic. is coming from the blog website [Mask R-CNN](#))

## 2.2.4 The image sharpness recognition

### Method:

- Make a convolution operation on a channel in the image ( generally using the gray value) through a *Laplace mask* [ *edge detection* ]
- Then calculate the standard deviation / variance,
- If the image has a higher variance, then it has a wider frequency response range, representing a normal, accurately focused image, the image is very clear.
- But if the image has a small variance, then it has a narrow frequency response range, which means that there are few edges in the image, and the image is very blurred.

- Set the appropriate threshold

### Explanation:

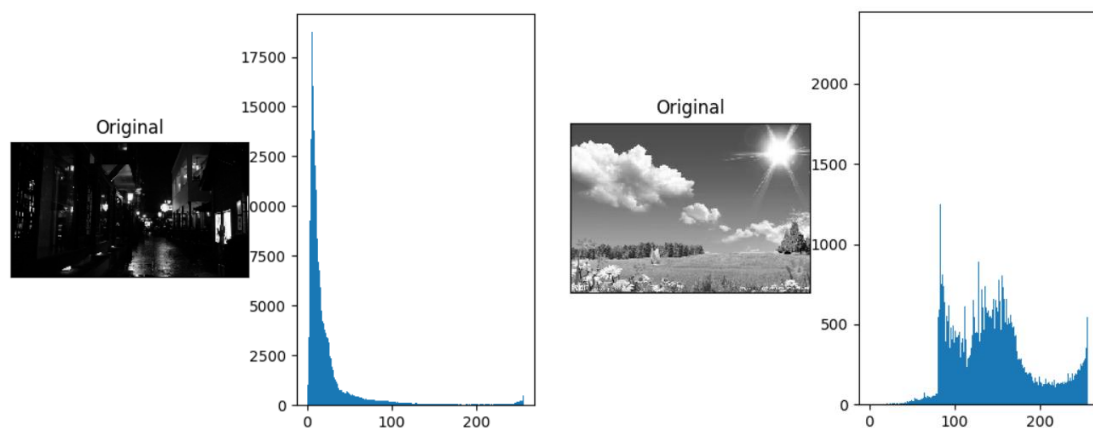
Blur in an image is caused by a lot of factors like defocus, camera shake, motion , etc.  
The most direct method to quantify the quality of blurred images is to calculate the *fast Fourier transform* of the image, and then view the high and low frequency distribution. If the image has a small amount of high-frequency components, the image can be considered blurry. However, it is very difficult to distinguish the specific threshold value of the high frequency amount, and an inappropriate threshold value will lead to extremely poor results.

## 2.2.5 The image brightness recognition

### Method:

- Obtain the grayscale histogram of the image → Judge according to the distribution of grayscale values
- Dark image gray value: concentrated in the front, such as around 0-30
- Bright image gray value: concentrated in the back
- Need to count the number of dark pixels, and then divide by the total number of image pixels to get the percentage  $p$
- Set the threshold of  $p$

### Explanation:



(pic. is coming from the blog website [Brightness](#) )

## 3. Results

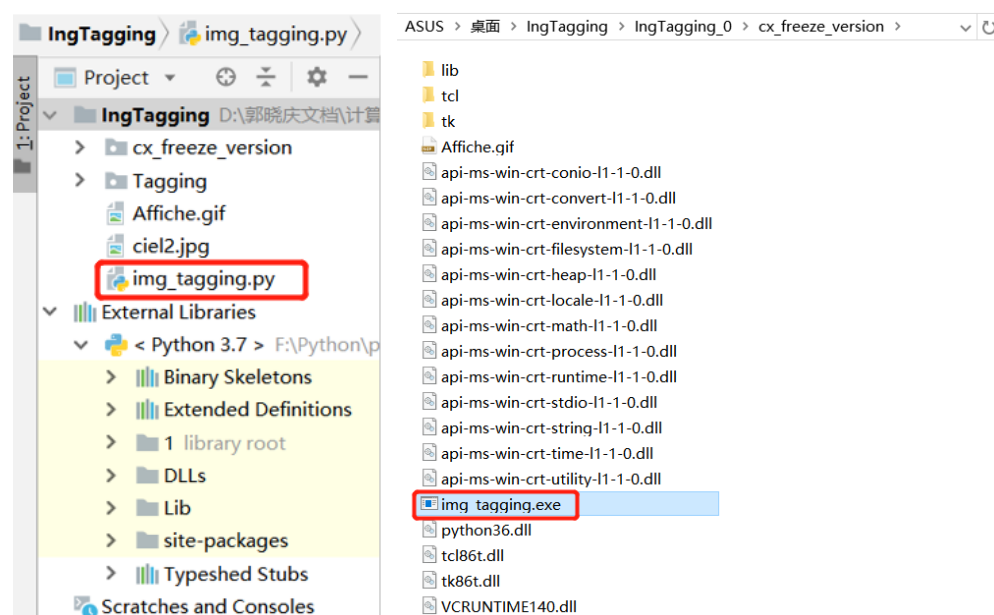
### 3.1 Original project operation and result display

[ Note: Introduction → Summary → background → TP3/Q6 ]

#### Step\_1: run

Open *IngTagging* project with Python and run “*img\_tagging.py*”.

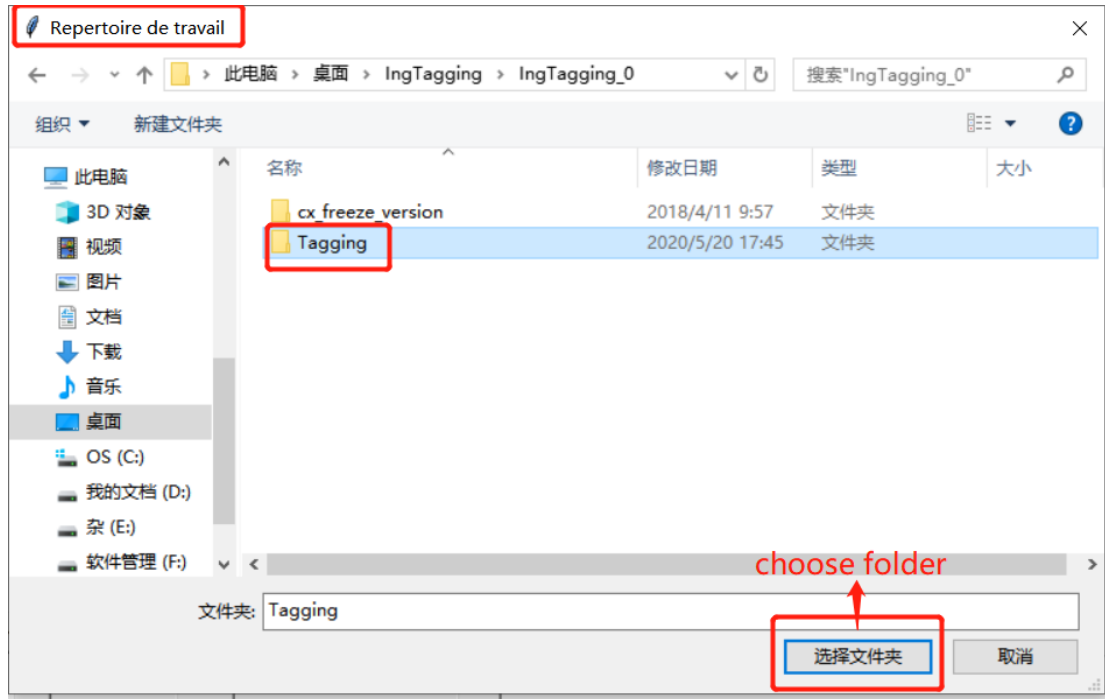
Or open the “*/IngTagging/cx\_freeze\_version*” and run “*img\_tagging.exe*”.



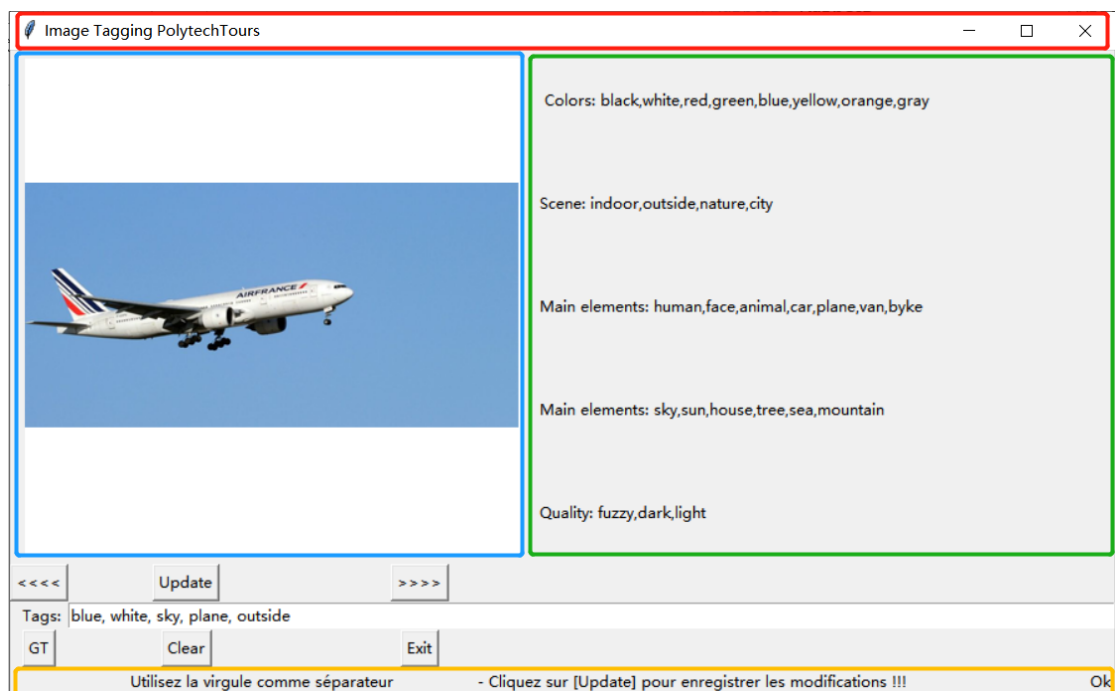
#### Step\_2: choose a folder of images which need to be tested

A folder selection window will pop up, then select a folder of images which need to test ( Here: the folder “*/Tagging*” in this case → Set working path ).





### Step\_3: Software Interface display



### Layout explanation:

- **Red frame:** Software system heading.
- **Blue frame:** Image display area under the images working directory.
- **Green frame:** The image attributes expected to be detected. And there are 5 parts:

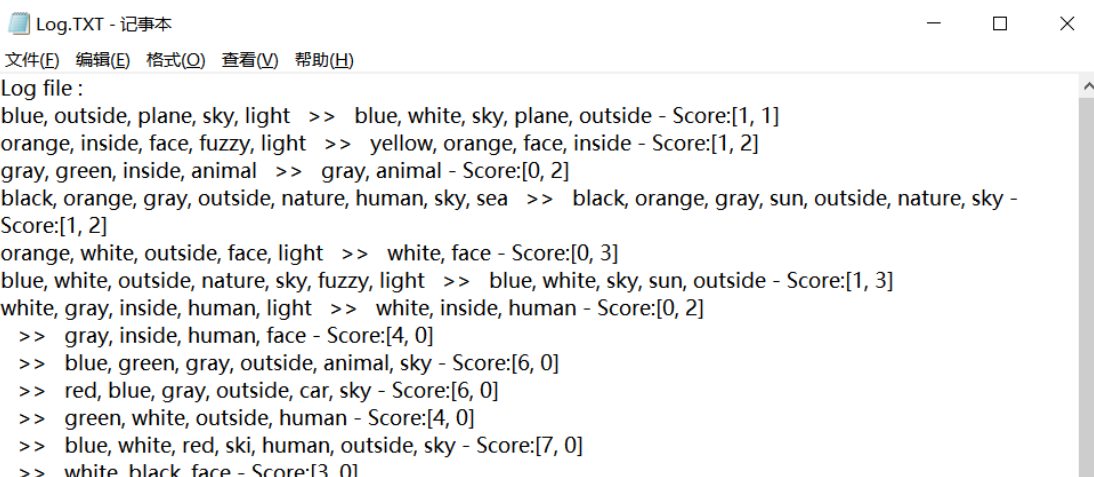
- Colors: black , white , red , green , blue , yellow , orange , gray , purple
- Scene: inside , outside , nature , city
- Main elements [ 1 – main objects detection ]: human , face , animal , car , plane , van , bike , ski
- Main elements [ 2 – main scene prediction ]: sky , sun , house , tree , sea , mountain , snow
- Quality: fuzzy , dark , light
- **Yellow frame:** Operation / status / image name prompt bar.

### Button function description:

- **Button [ <<<< ]:** Switch to the previous image.
- **Button [ *Update* ]:** Change the content of [ *Tags* ] bar and click on [ *Update* ] to save the modifications.
- **Button [ >>>> ]:** Switch to the next image.
- **Bar [ *Tags* ]:** Read the content of the txt file with the same name as the image in the subfolder “ */GT* ” and show it in bar.
- **Button [ *GT* ]:** Compare the content of the txt file with the same name of the image under the working directory of the current image with the content of the txt file with the same name of the image in the subfolder “ */GT* ” , and write the comparison result in the LOG.TXT file in the subfolder “ */GT* ”.
- **Button [ *Clear* ]:** Clear the content of [ *Tags* ] bar.
- **Button [ *Exit* ]:** Exit the software operating system.

## Step\_4: LOG.TXT file display

Evaluate software detection efficiency by *score*.



```

Log file :
blue, outside, plane, sky, light >> blue, white, sky, plane, outside - Score:[1, 1]
orange, inside, face, fuzzy, light >> yellow, orange, face, inside - Score:[1, 2]
gray, green, inside, animal >> gray, animal - Score:[0, 2]
black, orange, gray, outside, nature, human, sky, sea >> black, orange, gray, sun, outside, nature, sky -
Score:[1, 2]
orange, white, outside, face, light >> white, face - Score:[0, 3]
blue, white, outside, nature, sky, fuzzy, light >> blue, white, sky, sun, outside - Score:[1, 3]
white, gray, inside, human, light >> white, inside, human - Score:[0, 2]
>> gray, inside, human, face - Score:[4, 0]
>> blue, green, gray, outside, animal, sky - Score:[6, 0]
>> red, blue, gray, outside, car, sky - Score:[6, 0]
>> green, white, outside, human - Score:[4, 0]
>> blue, white, red, ski, human, outside, sky - Score:[7, 0]
>> white, black, face - Score:[3, 0]

```

```
>> outside, ski, blue, white, human - Score:[5, 0]
>> outside, yellow, green, house, nature, sky - Score:[6, 0]
>> gray, byke, human, city, outside - Score:[5, 0]
>> blue, outside, sea, sky, city, dark - Score:[6, 0]
>> outside, city, dark - Score:[3, 0]
>> - Score:[0, 0]
>> red, outside, car, gray - Score:[4, 0]
>> green, outside, car, gray - Score:[4, 0]
Final score : [262, 15] on 66 images
```

### Explanation:

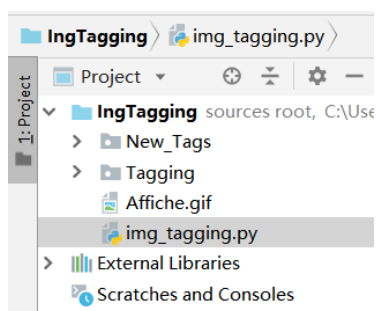
- `s1 >> s2 - Score:[ set1, set2]`  
`set1 = s1 - s2` # missing keywords  
`set2 = s2 - s1` # extra keywords
- Final score: [SET1, SET2]  
 SET1: the sum of all set1  
 SET2: the sum of all set1

## 3.2 New project operation and result display

[ Note: Introduction → Summary → New project description and goal → Continuation of exercise 6 of TP3 image analysis ]

### Step\_1: run

Open *IngTagging* project with Python and run “*img\_tagging.py*”.

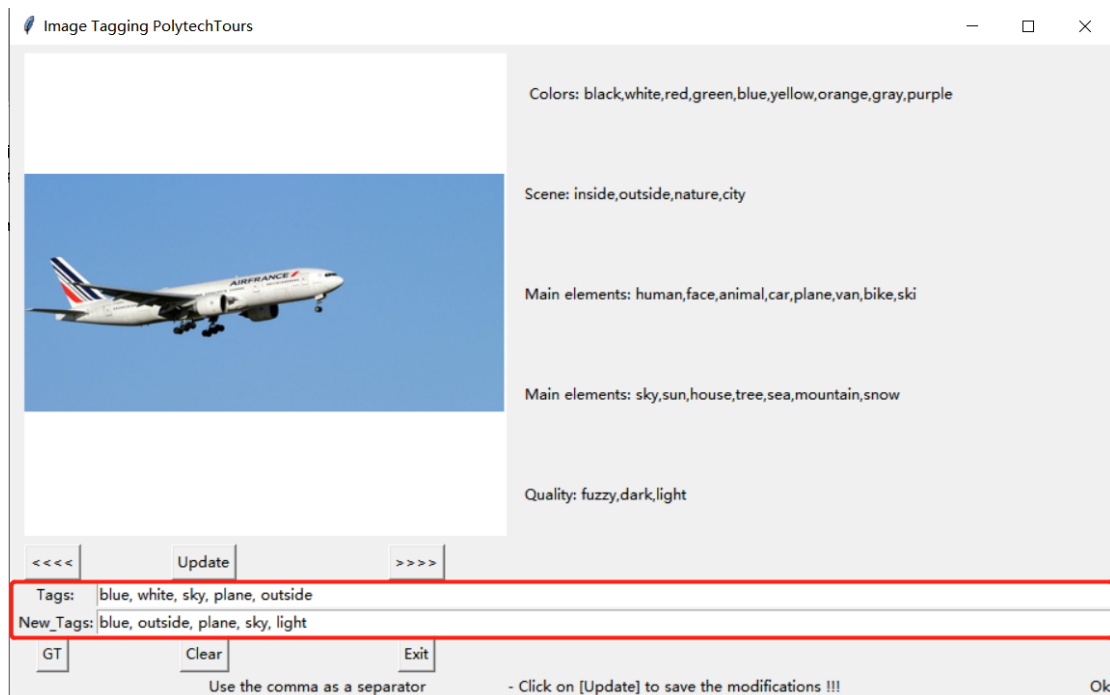


### Step\_2: choose a folder of images which need to test

Same as *Step\_2 of 3.1*

### Step\_3: Software Interface display

Almost same as *Step\_3 of 3.1*



## Addition:

### Button function description:

- **Bar [ *New\_Tags* ]:** Obtain image attribute labels detected by automatic image labeling algorithms , and write the new tags to the txt file with the same name of the image in the subfolder “ */NT* ”.
- **Button [ *GT* ]:** Compare the content of the txt file with the same name of the image in the subfolder “ */NT* ” with the content of the txt file with the same name of the image in the subfolder “ */GT* ” , and write the comparison result to the LOG.TXT file under the working directory of the current image.

## Step\_4: Extra effects - Masks / Renderings

### Main\_Elements [ 1 ] - The new image with the detection boxes and masks:

File [ */New\_Tags/Main\_Elements\_1.py* ]: Uncomment the part of red frame to generate and show the new image with the detection boxes and masks.

```

163 net.setInput(blob)
164
165 # Run the forward pass to get output from the output layers
166 boxes, masks = net.forward(['detection_out_final', 'detection_masks'])
167
168 # Extract the bounding box and mask for each of the detected objects
169 postprocess(boxes, masks, img)
170
171 # Uncomment these lines below to generate and show the img with the detection boxes and masks
172 # # Put efficiency information.
173 # t, _ = net.getPerfProfile()
174 # label = 'Mask-RCNN on 2.5 GHz Intel Core i7 CPU, Inference time for a img : %0.0f ms' % abs(t * 1000.0)
175 # cv.putText(img, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))
176 # # Write the new image
177 # if (image_path):
178 #     outputFile = image_path[:-4] + '_mask_rcnn_out.py.jpg'
179 #     cv.imwrite(outputFile, img.astype(np.uint8))
180 # img = cv.imread(outputFile)
181 # cv.imshow("OutputFile", img)
182
183 result = list(set(result_ME))
184 return result

```

Ex [ *airfrance\_4.jpg* ]:



## Main\_Elements [ 2 ] - Heatmap:

File [ */New\_Tags/Scene\_Main\_Elements\_1\_2.py* ]: Uncomment the part of red frame to generate and show the image with the informative region for predicting the category.

```

193
194 # generate class activation mapping
195 CAMs = returnCAM(features_blobs[0], weight_softmax, [idx[0]])
196
197 # Uncomment these lines below to generate and show the img with the informative region for predicting the
198 # # render the CAM and output
199 # img = cv2.imread(image_path)
200 # height, width, _ = img.shape
201 # heatmap = cv2.applyColorMap(cv2.resize(CAMs[0], (width, height)), cv2.COLORMAP_JET)
202 # result = heatmap * 0.4 + img * 0.5
203 # Informative_region_img = image_path.split('.')[0] + '_SceneRegion.jpg'
204 # cv2.imwrite(Informative_region_img, result)
205 # img = cv2.imread(Informative_region_img)
206 # cv2.imshow("show", img)
207
208 snow_list = ['snow', 'mountain_snowy']
209 ski_list = ['ski_slope']
210
211 nature_list = ['natural'] # + outside
212 city_list = ['cities', 'canal/urban', 'downtown', 'skyscraper', 'harbor', 'crosswalk', 'bus_station/ind
213
214 face_list = ['beauty_salon', 'arena/performance']

```

Ex [ *bougie.jpg* ]:



## Step\_5: LOG.TXT file display

Same as *Step\_4 of 3.1*

## 4. Follow-up

- 1) Obtain the main colors of the image

**Plan:**

- **Colors:** black , white , red , green , blue , yellow , orange , gray , purple

**Status and Problems:**

- completed

- 2) Obtain the main scene ( type of environment ) of the image

**Plan:**

- **Scene:** inside , outside , nature , city

**Status and Problems:**

- completed
- “ inside ” and “ city ” are sometimes inaccurately recognized.

- 3) Obtain the main objects of the image

**Plan:**

- **Main elements [ 1 – main objects detection ]:** human , face , animal , car , plane , van , bike , ski

**Status and Problems:**

- completed
- “ human ” and “ face ” are sometimes confused.
- It takes too long to call the pre-trained Mask\_RCNN model of Tensorflow for each image for training , and I have tried to improve the efficiency but it has no effect.

- 4) Make special scene prediction of the image

**Plan:**

- **Main elements [ 2 – special scene prediction ]:** sky , sun , house , tree , sea , mountain , snow

**Status and Problems:**

- almost completed
- “ sun ” can’t be recognized. → I don't know how to solve this problem.
- “ sky ” and “ sea ” are sometimes inaccurately recognized.
- “ house ” is sometimes inaccurately recognized.  
( Especially when there are many large buildings (E.g: skyscrapers ) in the image or the image is the inside part of the house. )

5) Detect the sharpness and brightness of the image

**Plan:**

- **Quality:** fuzzy , dark , light

**Status and Problems:**

- completed
- “fuzzy” is sometimes inaccurately recognized.  
( Especially for the focused images / the focused photos. )

6) Generate a new tags txt file

**Plan:**

- Generate a txt file with the same name of image

**Status and Problems:**

- completed

7) Compare two tags txt files and Assess the accuracy of detection

**Plan:**

- Compare the new\_tags txt file in the “./NT ” folder with the same name txt file in the “./GT ” folder and write the comparison result to the LOG.TXT file under the working directory of the current image

**Status and Problems:**

- completed





## Conclusion

This is a very challenging project, which can not only exercise my computer skills, but also improve my English communication skills and writing skills. This project involves the field of deep learning-computer vision. I took the AI basic course this semester, but because I did not fully master the course, I felt a bit hard when I started this project.

The hardest part of the project is object detection and scene recognition. I set up a step-by-step study plan to learn the necessary parts of the project. In addition, I selected several datasets and tried to use the convolutional network to train the model by myself in order to master the basic knowledge of object detection.

I encountered many difficulties during the process of the project. Although I tried to solve them one by one until the project was completed, there were still some problems that I could not solve, for example: "sun" tag could not be detected; The object detection is a time-consuming operation which results in the inefficient system operation. But I found that I am very interested in this field, and then I will continue to study this field.

Finally, dear *Mr. Ramel* and *Miss Bachet*, I will express my great thanks and wishes to you. Thank you very much for your help in this project.



# Bibliography

- [1] <https://www.learnopencv.com/deep-learning-based-object-detection-and-instance-segmentation-using-mask-r-cnn-in-opencv-python-c/>
- [2] <https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/>
- [3] <https://stackabuse.com/image-recognition-in-python-with-tensorflow-and-keras/>
- [4] <https://www.tensorflow.org/>
- [5] <https://machinelearningmastery.com/deep-learning-with-python/>
- [6] <https://en.wikipedia.org/wiki/OpenCV>
- [7] <https://towardsdatascience.com/the-state-of-ai-in-2020-1f95df336eb0>
- [8] <https://www.forbes.com/sites/mariyayao/2020/01/22/what-are--important-ai--machine-learning-trends-for-2020/#1977ce492323>
- [9] <http://places2.csail.mit.edu/index.html>



## Abstract

This project is a part of Polytech ORA. The purpose of ORA is to practice my English and computer skills. And the purpose of this project is, under the environment setup of combining *Python* and *OpenCV*, to build a software system to associate keywords (Contents: *image attributes, type of scene, objects, etc.*) with images or photos, and to compare results to assess the accuracy of detection. In addition, the main part of this project is to research and improve the related algorithms (Ex: image processing, object detection, instance segmentation and scene classification) in the area of image recognition of the domain Computer Vision of AI/Machine learning/Deep learning.

**Keywords:**

Deep learning, Object Detection, Instance Segmentation, Scene Classification, Mask R-CNN