香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

第四课 无人船定位

CUHK(SZ)

2019年7月

Contents 目录

01 机器视觉

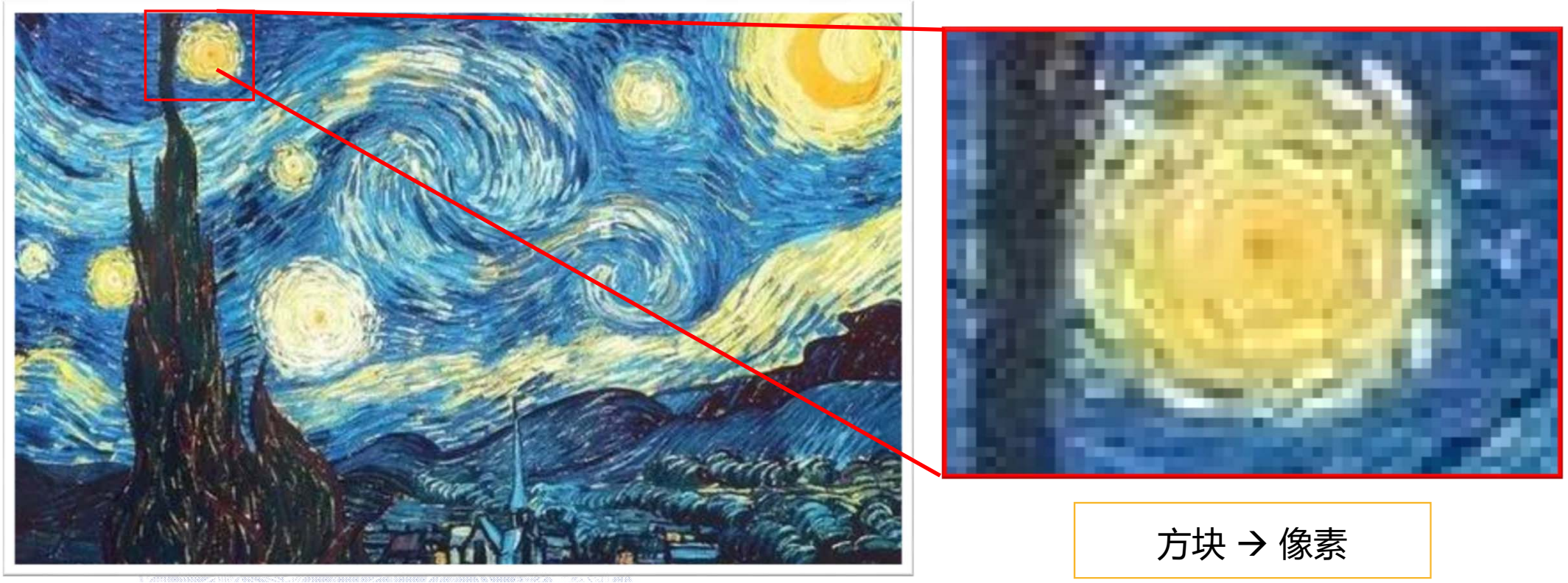
02 图像识别主要方法

03 YOLO 网络

04 实践：无人船定位

1.1 计算机眼中的图片

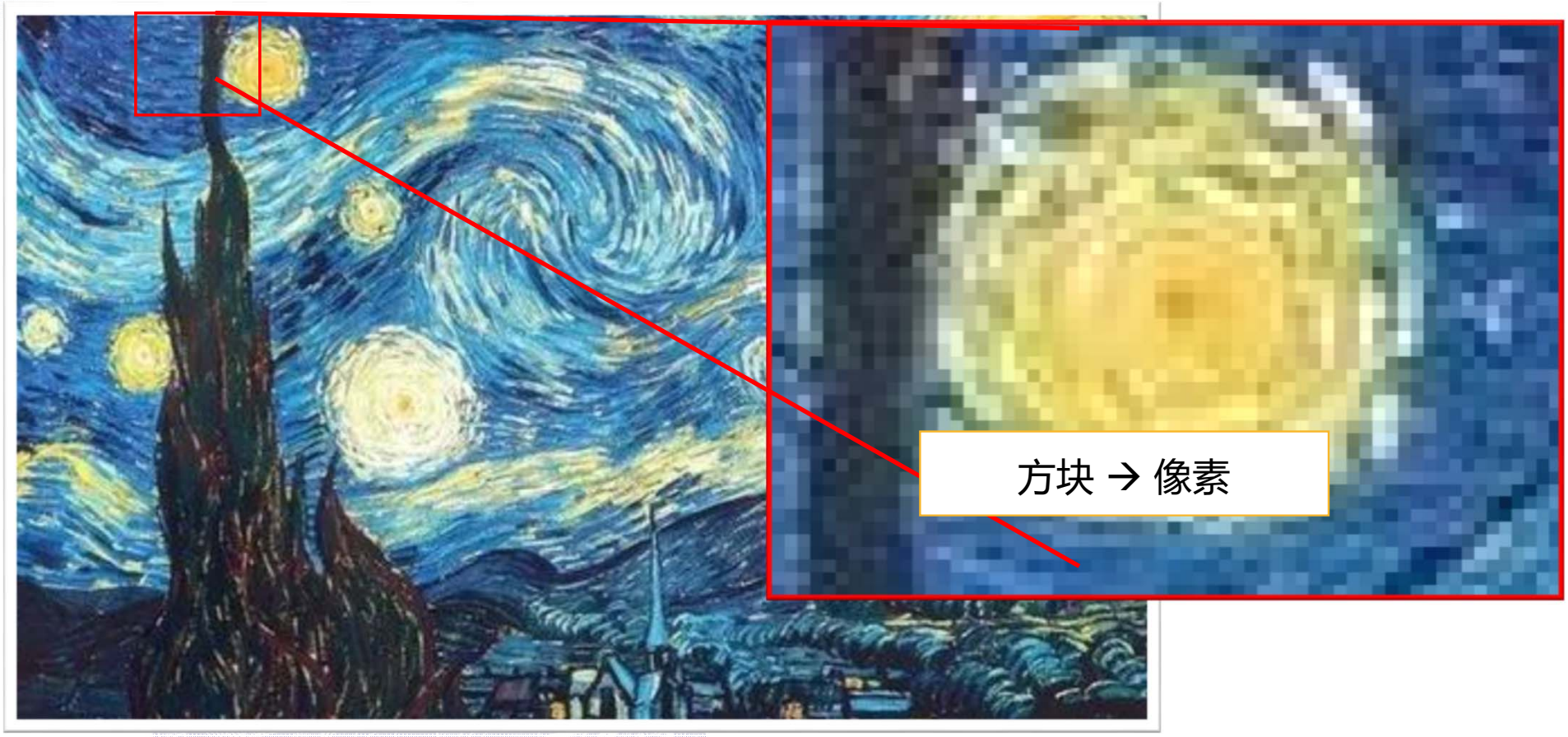
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen



方块 → 像素

像素是图像中不可分割的最小单位或元素，以一个单一颜色的小格存在;

像素明确的位置和被分配的色彩数值决定图像在屏幕上所呈现的样子。



像素是图像中不可分割的最小单位或元素，以一个单一颜色的小格存在;

像素明确的位置和被分配的色彩数值决定图像在屏幕上所呈现的样子。

1.1计算机眼中的图片

每一个点阵图像包含了一定量的像素，这些像素明确的位置和被分配的色彩数值决定图像在屏幕上所呈现的样子。

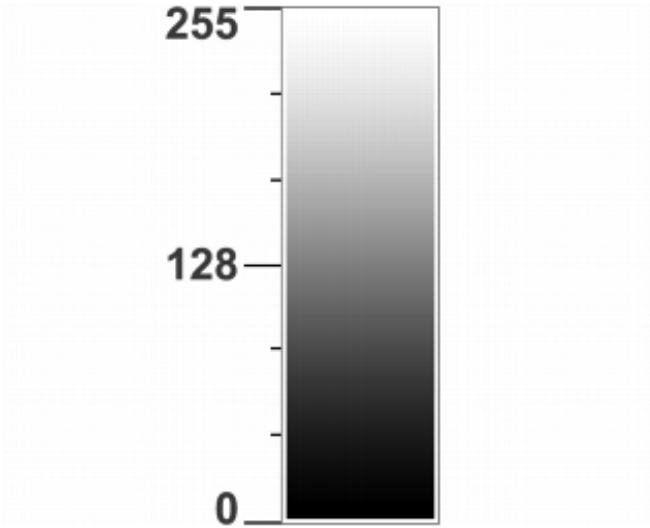
查看图片详细信息，可以看到图片分辨率、宽度、高度等数值。

图像的宽度、高度即为图像在宽度和高度方向的像素个数，宽度X高度即为图像的分辨率。

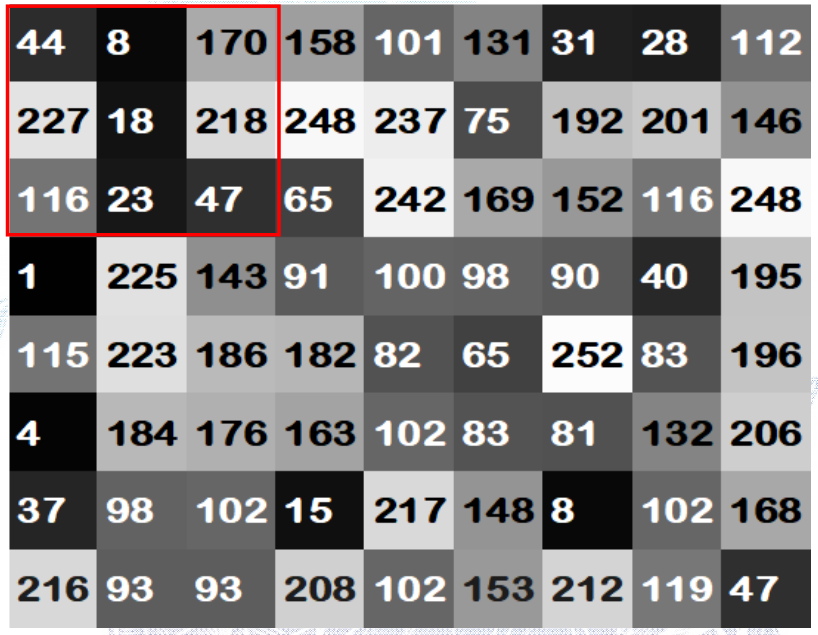
1.2 像素的表示

黑白图，也即灰度图，常见的卫星图像、航空照片、许多地球物理观测数据等都会以灰度图呈现。计算机中一般讲灰度分为256阶（下图仅分为21阶，计算机中分得更细），分别用（0-255）表示其值，即灰度值。因此黑白照片中，每个像素点可以用依据它的灰度，用对应的灰度值表示。

21阶灰度



1.2 像素的表示



图中红框内灰度矩阵表示：

$$\begin{bmatrix} 44 & 8 & 170 \\ 227 & 18 & 218 \\ 116 & 23 & 47 \end{bmatrix}$$

相应地，整张黑白图片的表示也类似。

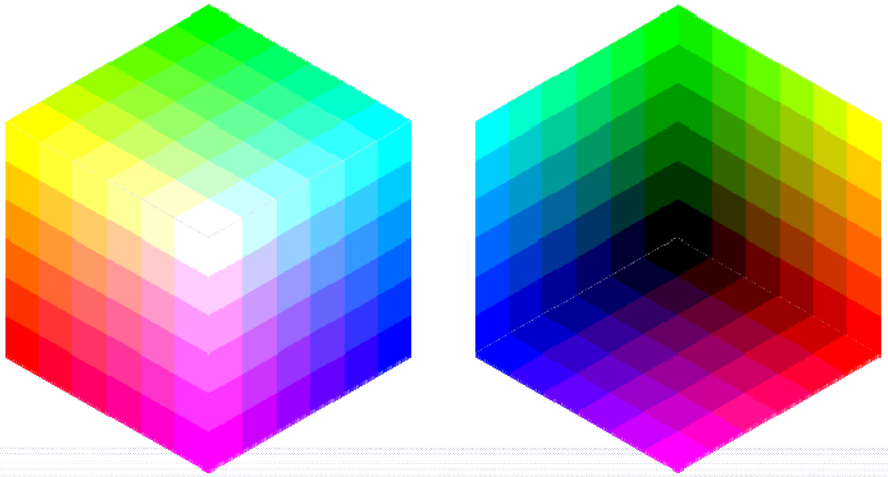
1.2 像素的表示

彩色图片，也叫RGB图片。R(red)即红色，G（green）即绿色，B（blue）即蓝色。红、绿、蓝是不能再分解的三种基本颜色，也叫**三原色**。绘画中，调节三原色的比例可以混合出各种各样的颜色。因此，科学家也很聪明地利用三原色为计算机中的图片“着色”：通过设置三个变量，分别代表红、绿、蓝，并选择不同的数值来表示混合量，即可“调出”我们所需要的颜色。

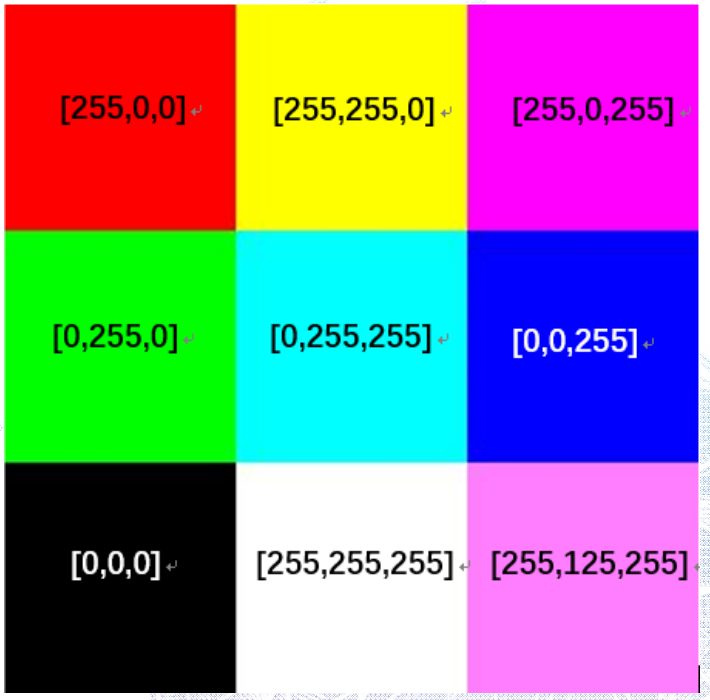
数值的选择上，如同灰度值，将红、绿、蓝各分成256阶，分别用（0-255）表示其不同程度。因此每个颜色都有对应的RGB值，如：

红：255，0，0
绿：0，255，0
蓝：0，0，255

其他颜色则可以由三原色混合而来，如黄色是用红色和绿色调和出来的，所以其RGB值为 255，255，0。



1.2 像素的表示

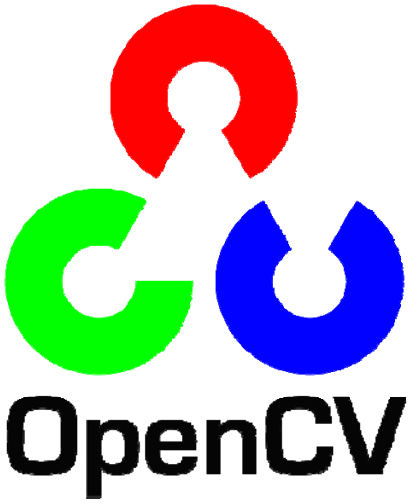


将每一个原色作为一个通道，彩色图片即具有三个通道。
如图3x3颜色块，对应R，G，B三个通道矩阵分别为：

$$\begin{bmatrix} 255 & 255 & 0 \\ 0 & 255 & 0 \\ 0 & 255 & 255 \end{bmatrix} \quad \begin{bmatrix} 0 & 255 & 0 \\ 255 & 255 & 0 \\ 0 & 255 & 125 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 255 \\ 0 & 255 & 255 \\ 0 & 255 & 255 \end{bmatrix}$$

1.3 计算机视觉库OpenCV

人工智能视觉处理一般会用一个重要的计算机视觉库：OpenCV。
OpenCV全称：Open Source Computer Vision Library。它是一个开源(源代码公开)的计算机视觉库，应用非常广泛，包括物体识别、人脸识别、动作识别、人机互动、运动跟踪、无人驾驶等等。OpenCV还为包括Python在内的多种编程语言提供了接口，可以在Python中直接调用OpenCV，实现图像处理和计算机视觉方面的很多通用算法。



Python是一种计算机程序设计语言，现在已经发展成为人工智能首选语言。
它的主要特点是：1、简单易学；2、丰富的包和库，可以直接调用。

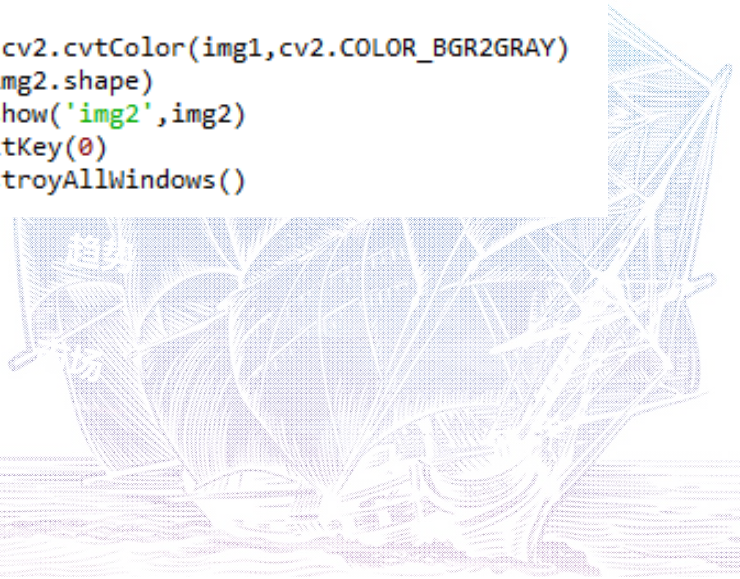
1.3 计算机视觉库OpenCV

演示1：在Python中利用OpenCV库实现图像读取，显示，灰度化等功能（参见3.1.py中3.1段代码。）

```
import numpy as np
import cv2

#打开图片，并读取分辨率信息
img1 = cv2.imread('image.jpg')
print(img1.shape)
cv2.imshow('img1',img1)
cv2.waitKey(0)
cv2.destroyAllWindows()

img2 = cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
print(img2.shape)
cv2.imshow('img2',img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



1.3 计算机视觉库OpenCV

演示1：在Python中利用OpenCV库实现图像读取，显示，灰度化等功能（参见3.1.py中3.1段代码。）

函　数	说　明
cv2.imread()	读取图像。若图像在当前路径下（即图像与Python文件在同一个文件夹中），直接输入图像文件名即可。若图像与Python文件不在同一个目录，需给出完整的图像路径。
cv2.imShow()Output	在窗口中显示图像。该窗口和图像的原始大小自适应（自动调整到原始尺寸）。第一个参数是一个窗口名称，它是一个字符串类型。可以创建任意数量的窗口，但必须使用不同的窗口名称。第二个参数是要显示的图像。
cv2.waitKey(0)	是一个和键盘绑定的函数，它的作用是等待一个键盘的输入。如果没有这个函数，创建的图片窗口会闪一下就消失了。若希望图片窗口能停留输出，即需要使用该函数。参数0表示按下键盘0毫秒后，执行下一语句。
cv2.destroyALLWindows()	销毁创建的所有窗口。如果要销毁某特定窗口，需使用函数cv2.destroyWindow()，其中传递确切的窗口名称作为参数。
cv2.cvtColor()	色彩空间转化函数。第一个参数给出需要转化色彩空间的图像，第二个参数写出转化方式。cv2.COLOR_BGR2GRAY即将图像从BGR转化为灰度图。

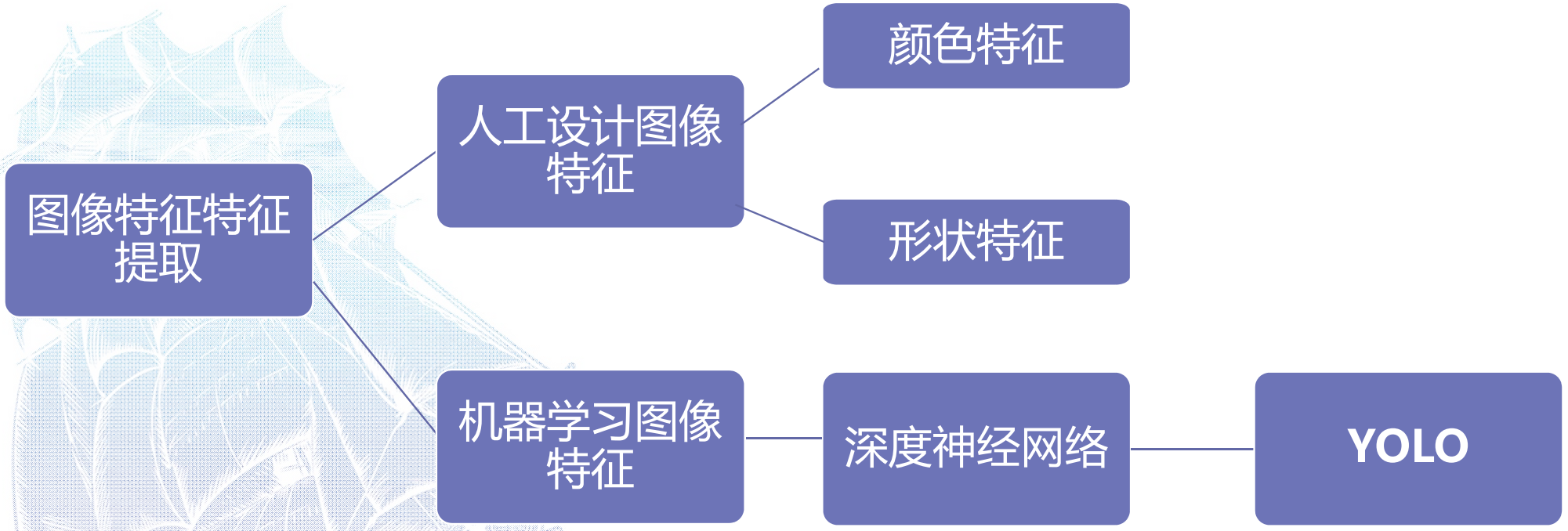
01 机器视觉

02 图像识别主要方法

03 YOLO 网络

04 实践：无人船定位

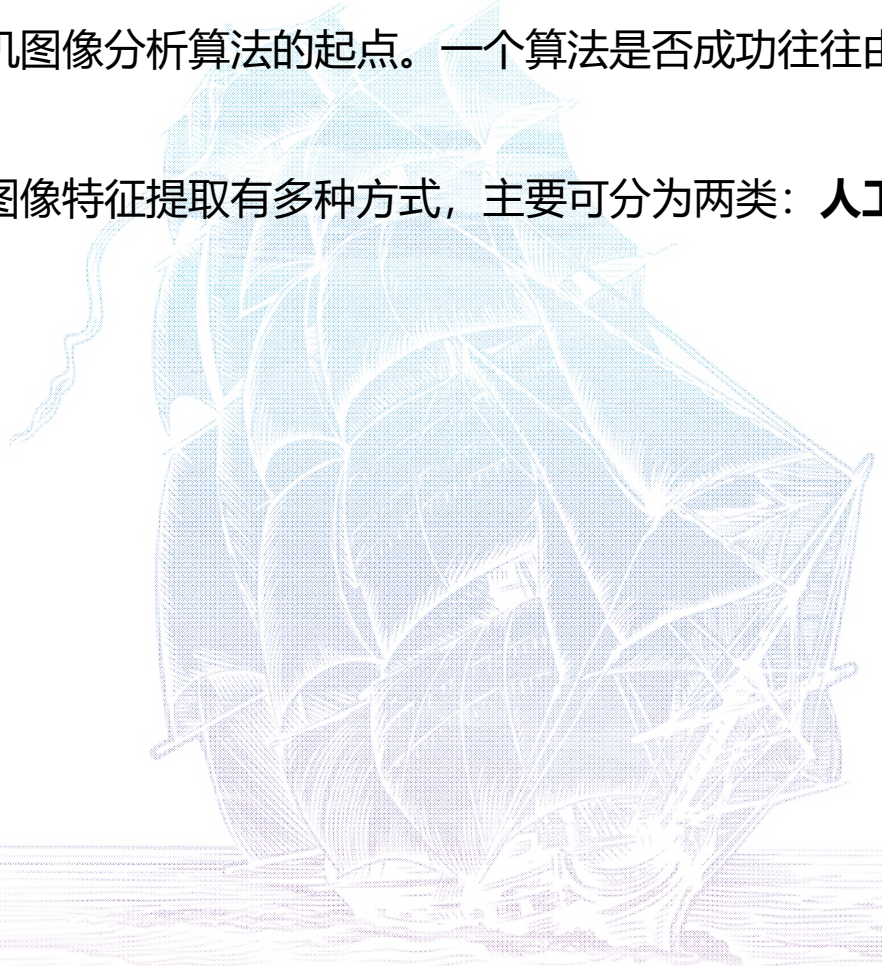
2. 图像识别的主要方法



2.1 图像特征提取

计算机识别物体首先需要对其**特征**进行提取。特征是一个数字图像中“有趣”的部分，它是许多计算机图像分析算法的起点。一个算法是否成功往往由它使用和定义的特征决定。

图像特征提取有多种方式，主要可分为两类：**人工设计图像特征**和**机器学习图像特征。**



2.1 图像特征提取

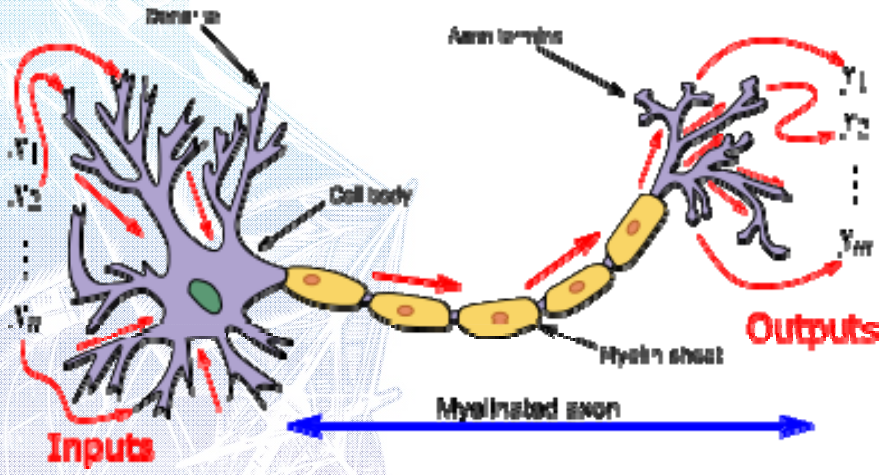
人工设计图像特征：比如根据图像的**颜色特征**、**纹理特征**、**形状特征**和**空间关系特征**等，**人为**的设计一些图像特征，用于不同类别图像分类。不同的特征提取会通过不同的数学方式去实现。比如图像直方图是一种可以用于图像颜色特征提取的数学方式。
特点：稳定，快速，适用于特定环境、特定特征。

人工设计形状特征

机器学习图像特征：利用大数据和深度神经网络，让**机器自己学习**到图像特征。
特点：适用广，复杂。

2.2 人工神经网络

人工神经网络（Artificial Neural Network，ANN），简称神经网络（Neural Network，NN），在机器学习和认知科学领域，是一种模仿生物神经网络（动物的中枢神经系统，特别是大脑）的结构和功能的数学模型或计算模型，用于对函数进行估计或近似。
对人类中枢神经系统的观察启发了人工神经网络这个概念。在人工神经网络中，简单的人工节点，称作神经元（neurons），连接在一起形成一个类似生物神经网络的**网状结构**。

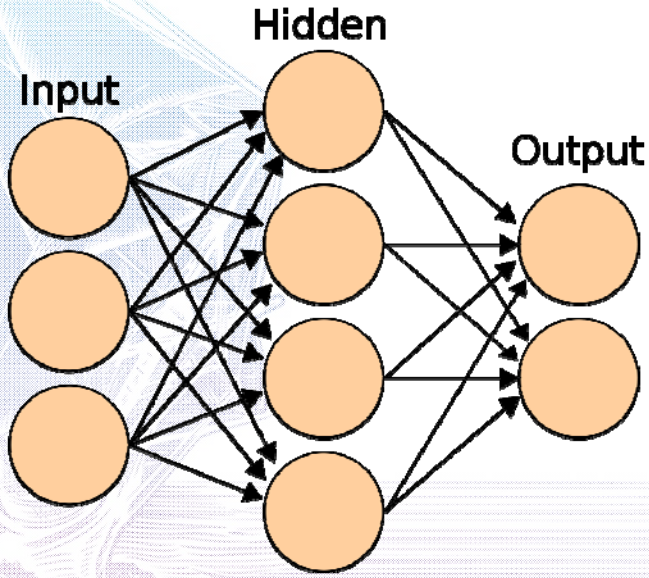


2.2 人工神经网络

输入层（Input layer），众多神经元（Neuron）接受大量非线形输入消息。输入的消息称为输入向量。

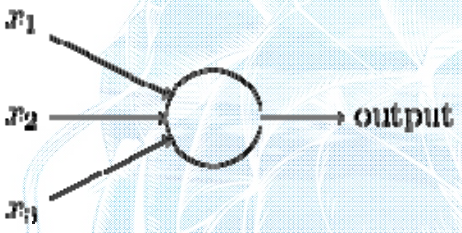
输出层（Output layer），消息在神经元链接中传输、分析、权衡，形成输出结果。输出的消息称为输出向量。

隐藏层（Hidden layer），简称“隐层”，是输入层和输出层之间众多神经元和链接组成的各个层面。隐层可以有一层或多层。隐层的节点（神经元）数目不定，但数目越多神经网络的非线性越显著，从而神经网络的强健性（robustness）（控制系统在一定结构、大小等的参数摄动下，维持某些性能的特性）更显著。习惯上会选输入节点1.2至1.5倍的节点。



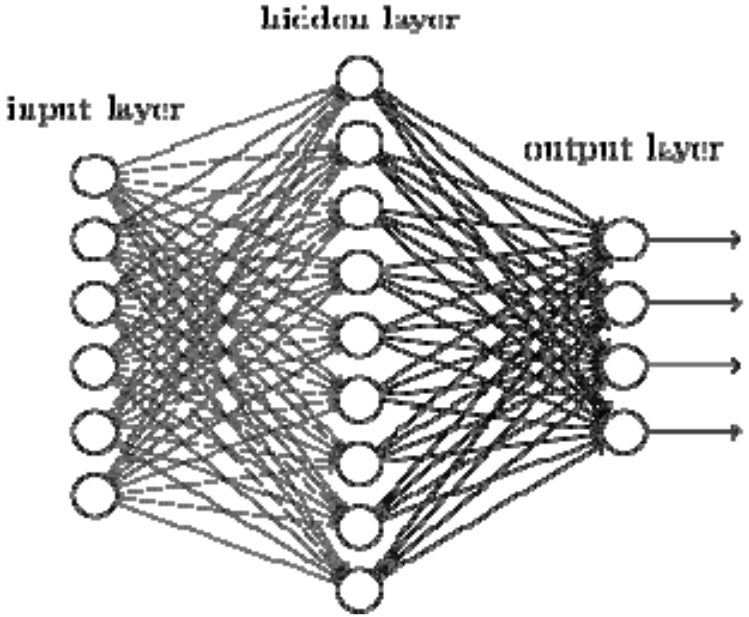
2.2 人工神经网络

① 20世纪五、六十年代，单层感知器



特点：只有输入层、输出层，没有隐含层，是最简单的神经网络，可以解决分类问题。

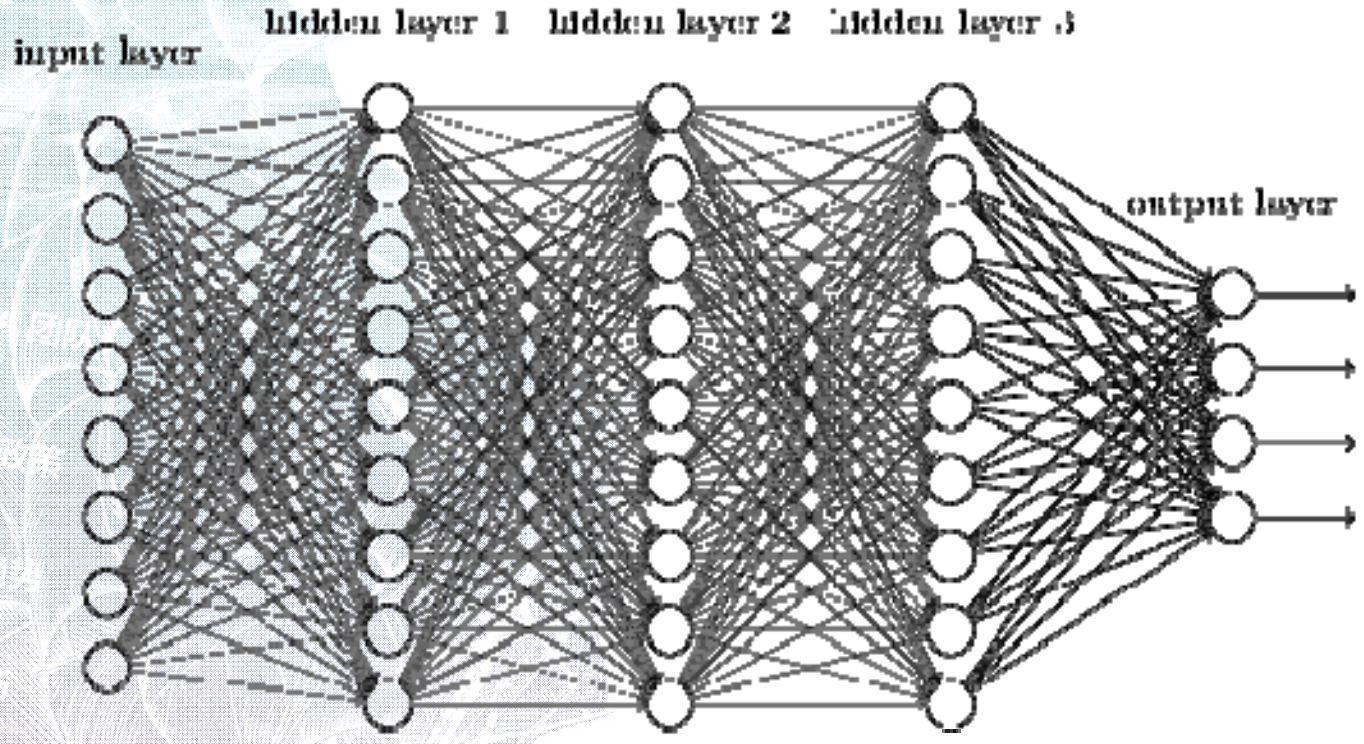
② 20世纪八十年代，多层感知器



特点：拥有输入层、输出层和一个隐含层，可以解决较复杂问题。

2.2 人工神经网络

③ 2006年，**深度神经网络**



特点：拥有输入层、输出层和多个隐含层，可以解决复杂问题。

01 机器视觉

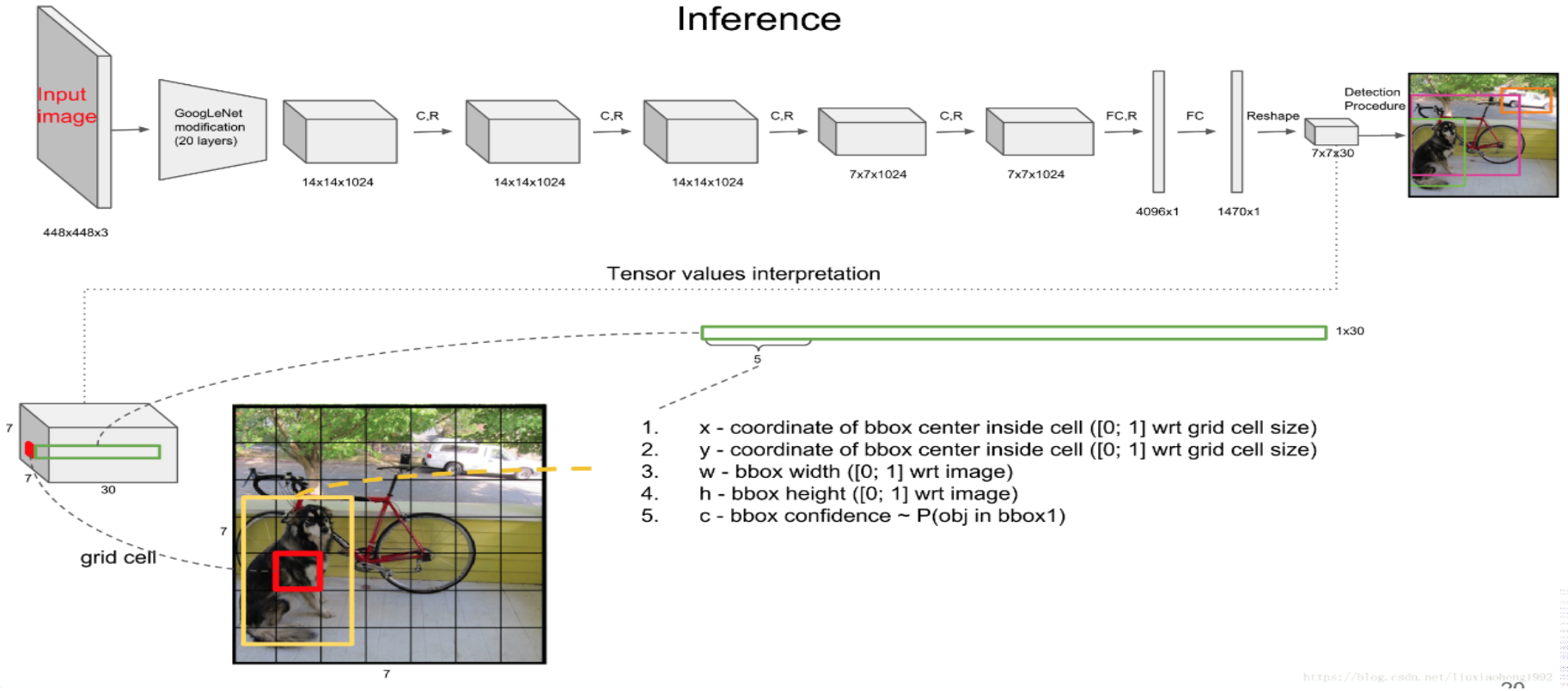
02 图像识别主要方法

03 YOLO 网络

04 实践：无人船定位

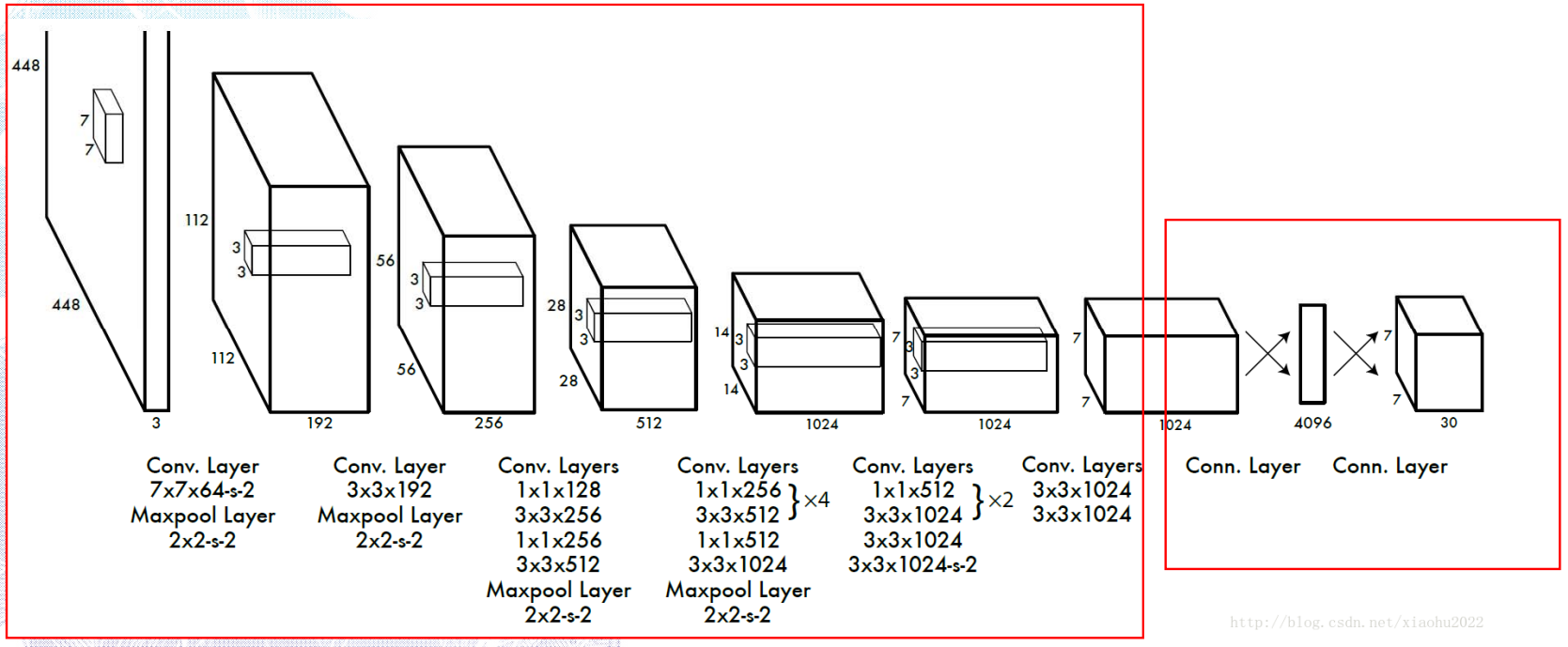
3.1 YOLO 网络介绍

YOLO全名You only look once（你只需要看一眼），是一个用于摄像头的实时目标检测系统。它能分辨出9000多种物体，且分辨快速，相比利用深度学习进行目标检测的开山之作R-CNN系统，能快上1000倍。



3.1 YOLO 网络介绍

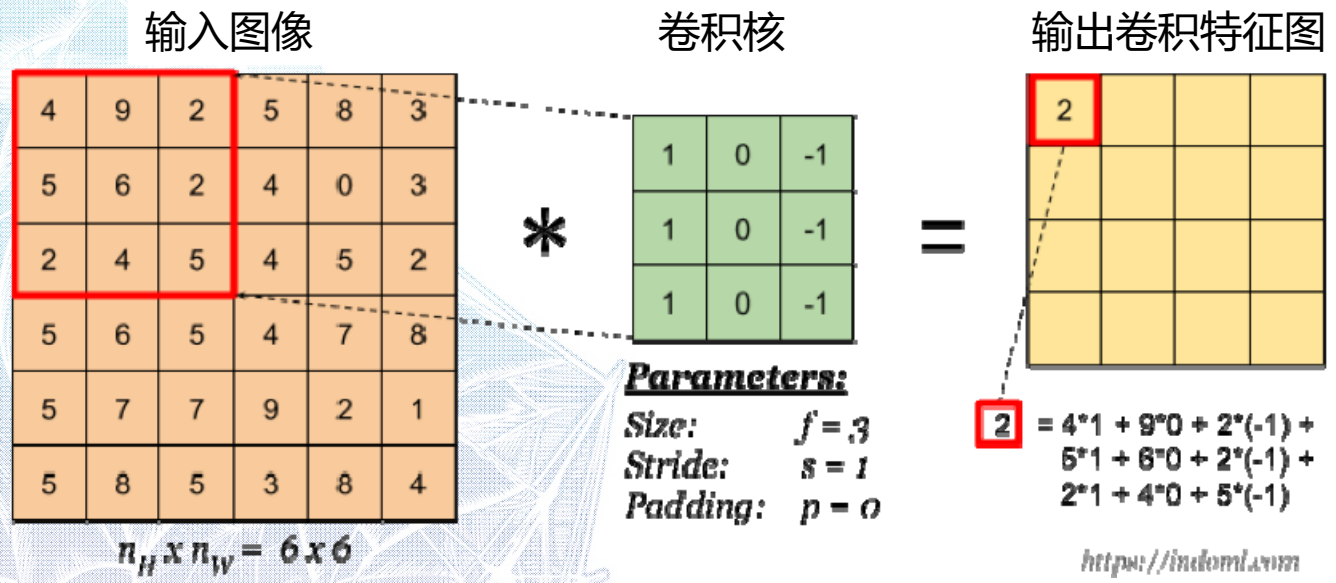
目前，YOLO有多个不同版本，tiny-YOLO, YOLOv1，YOLOv2，YOLOv3等，不同版本结构有所不同，但都包含多个**卷积层**和2个**全链接层**。



多个卷积层　　两个全链接层

3.2 卷积层

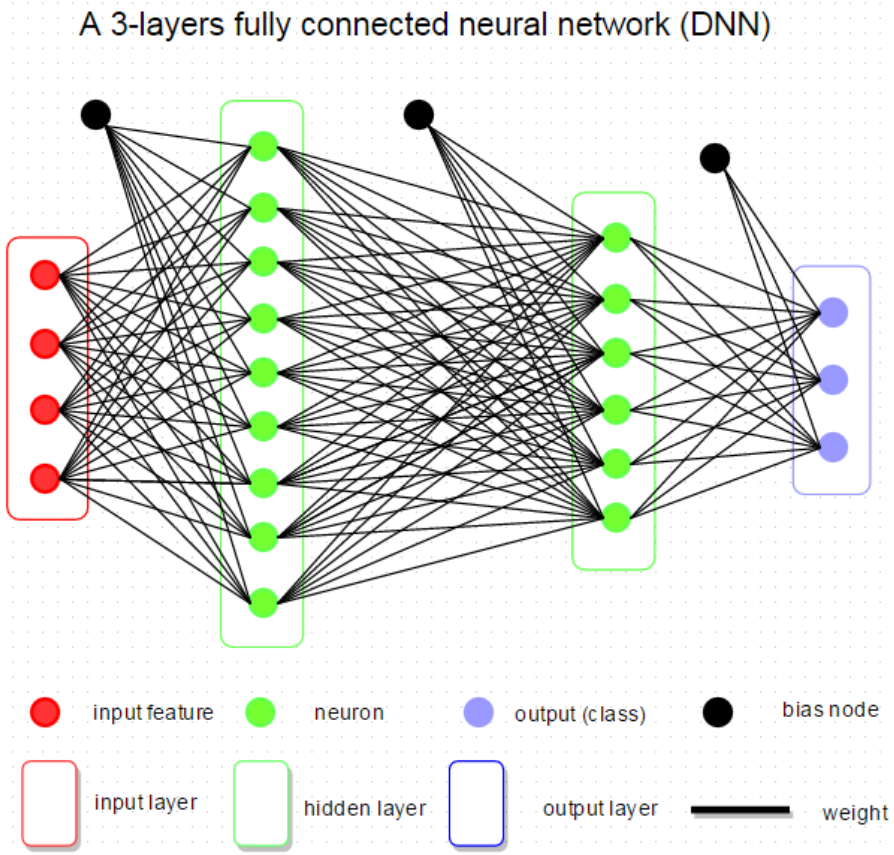
卷积是分析数学中一种重要的运算，YOLO网络中的卷积层计算思想如下



YOLO卷积层用来提取图像特征。 卷积核中的数值即是需要训练的权值系数。

3.2 全链接

全链接：即将每一个结点都与上一层的**所有结点**相连。



全链接层权重系数是最多的，每一条线即是一个权重系数。全链接层作用为用来预测图像位置和类别置信度一张图片，经过YOLO网络后，会给出要识别物体的中心坐标（x，y），宽度w，高度h，以及类别置信度c。这些工作都在全链接层进行。

01 机器视觉

02 图像识别主要方法

03 YOLO 网络

04 实践：无人船定位

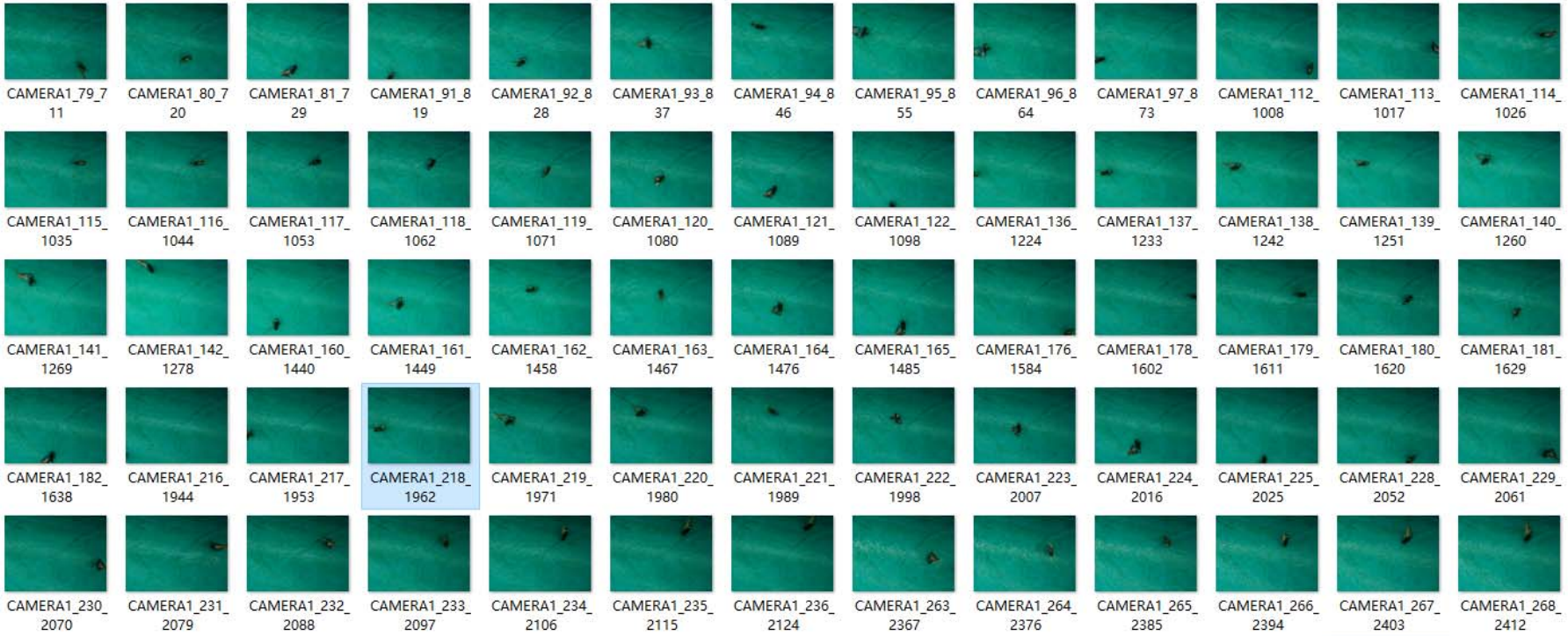
4 实践：无人船定位

虽然之前演示的YOLO网络，能够分辨出9000多种物体。但对我们无人船来说，有些冗余，且无人船环境与YOLO原本识别库图像差别较大，训练得到的权重系数适用性不强。所以针对我们的无人船场景，有必要自己进行一次训练，获得更有针对性的权重系数。因此，实现无人船视觉定位，需要经过：

4.1 训练

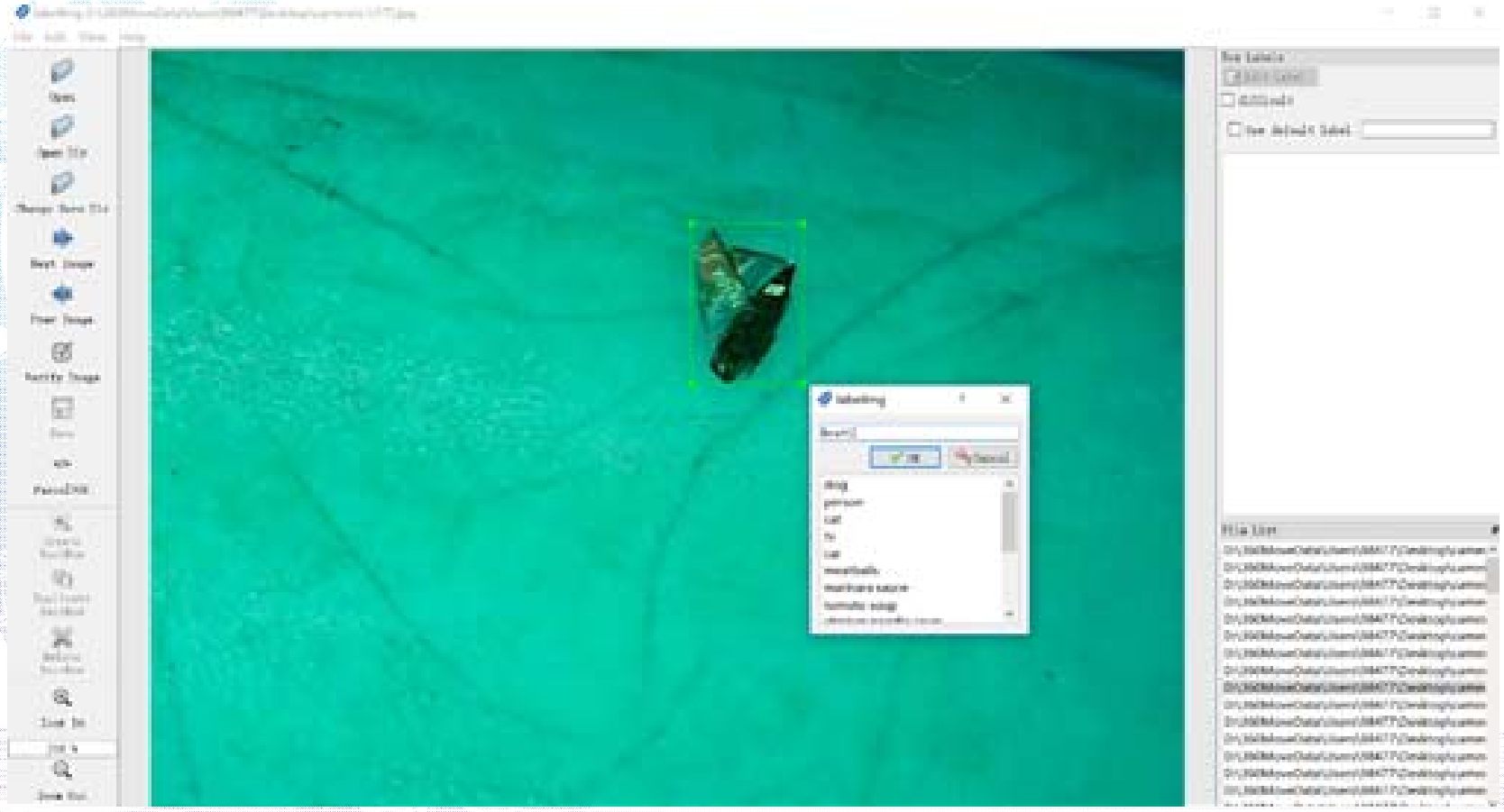
神经网络训练的目的是获得合适的**权重系数**。训练之前我们首先要准备好“素材”（无人船场景图片），供神经网络“学习”。

Step1：用摄像头拍摄大量拥有无人船的视频，并从视频从取出一帧帧图片。



4.1 训练

Step2：用lableImg软件给获得的图片打标签，制作训练数据集。(神经网络训练中最为耗时的步骤)



4.1 训练

Step3：运用YOLO网络对数据集进行训练，这里选的是tiny-YOLOv2。

```
Region Avg IOU: 0.410717, Class: 1.000000, Obj: 0.002305, No Obj: 0.002051, Avg Recall: 0.156250,  count: 32
131: 6.704353, 16.102207 avg, 0.004000 rate, 0.358082 seconds, 16768 images
Loaded: 0.000059 seconds
Region Avg IOU: 0.388631, Class: 1.000000, Obj: 0.002265, No Obj: 0.001985, Avg Recall: 0.031250,  count: 32
Region Avg IOU: 0.423133, Class: 1.000000, Obj: 0.002254, No Obj: 0.001985, Avg Recall: 0.281250,  count: 32
Region Avg IOU: 0.461764, Class: 1.000000, Obj: 0.002223, No Obj: 0.001985, Avg Recall: 0.250000,  count: 32
Region Avg IOU: 0.436554, Class: 1.000000, Obj: 0.002248, No Obj: 0.001985, Avg Recall: 0.187500,  count: 32
Syncing... Done!
144: 6.832729, 15.175260 avg, 0.004000 rate, 0.620359 seconds, 18432 images
Loaded: 0.000046 seconds
Region Avg IOU: 0.425193, Class: 1.000000, Obj: 0.002164, No Obj: 0.001927, Avg Recall: 0.218750,  count: 32
Region Avg IOU: 0.425633, Class: 1.000000, Obj: 0.002162, No Obj: 0.001927, Avg Recall: 0.218750,  count: 32
Region Avg IOU: 0.455220, Class: 1.000000, Obj: 0.002128, No Obj: 0.001928, Avg Recall: 0.250000,  count: 32
Region Avg IOU: 0.441174, Class: 1.000000, Obj: 0.002144, No Obj: 0.001928, Avg Recall: 0.250000,  count: 32
145: 6.963758, 14.354110 avg, 0.004000 rate, 0.380939 seconds, 18560 images
Loaded: 0.000055 seconds
Region Avg IOU: 0.438228, Class: 1.000000, Obj: 0.002146, No Obj: 0.001876, Avg Recall: 0.281250,  count: 32
Region Avg IOU: 0.446879, Class: 1.000000, Obj: 0.002144, No Obj: 0.001877, Avg Recall: 0.312500,  count: 32
Region Avg IOU: 0.428347, Class: 1.000000, Obj: 0.002122, No Obj: 0.001877, Avg Recall: 0.250000,  count: 32
Region Avg IOU: 0.425725, Class: 1.000000, Obj: 0.002113, No Obj: 0.001877, Avg Recall: 0.218750,  count: 32
146: 7.011239, 13.619823 avg, 0.004000 rate, 0.353885 seconds, 18688 images
Loaded: 0.000060 seconds
Region Avg IOU: 0.442809, Class: 1.000000, Obj: 0.002115, No Obj: 0.001832, Avg Recall: 0.250000,  count: 32
Region Avg IOU: 0.465820, Class: 1.000000, Obj: 0.002096, No Obj: 0.001832, Avg Recall: 0.312500,  count: 32
Region Avg IOU: 0.404827, Class: 1.000000, Obj: 0.002108, No Obj: 0.001832, Avg Recall: 0.218750,  count: 32
Region Avg IOU: 0.432866, Class: 1.000000, Obj: 0.002093, No Obj: 0.001832, Avg Recall: 0.156250,  count: 32
147: 6.975258, 12.955366 avg, 0.004000 rate, 0.354864 seconds, 18816 images
Loaded: 0.000048 seconds
Region Avg IOU: 0.425635, Class: 1.000000, Obj: 0.002055, No Obj: 0.001792, Avg Recall: 0.312500,  count: 32
Region Avg IOU: 0.435191, Class: 1.000000, Obj: 0.002065, No Obj: 0.001793, Avg Recall: 0.187500,  count: 32
Region Avg IOU: 0.437522, Class: 1.000000, Obj: 0.002059, No Obj: 0.001793, Avg Recall: 0.250000,  count: 32
Region Avg IOU: 0.438167, Class: 1.000000, Obj: 0.002069, No Obj: 0.001792, Avg Recall: 0.281250,  count: 32
Syncing... Done!
160: 7.029617, 12.362791 avg, 0.004000 rate, 0.585723 seconds, 20480 images
Resizing
352
Loaded: 0.000053 seconds
Region Avg IOU: 0.431402, Class: 1.000000, Obj: 0.002050, No Obj: 0.001758, Avg Recall: 0.187500,  count: 32
Region Avg IOU: 0.445471, Class: 1.000000, Obj: 0.002000, No Obj: 0.001759, Avg Recall: 0.343750,  count: 32
Region Avg IOU: 0.440954, Class: 1.000000, Obj: 0.002046, No Obj: 0.001759, Avg Recall: 0.343750,  count: 32
Region Avg IOU: 0.411415, Class: 1.000000, Obj: 0.002017, No Obj: 0.001758, Avg Recall: 0.250000,  count: 32
161: 6.794780, 11.805990 avg, 0.004000 rate, 0.265904 seconds, 20608 images
Loaded: 0.000042 seconds
Region Avg IOU: 0.459690, Class: 1.000000, Obj: 0.002018, No Obj: 0.001727, Avg Recall: 0.343750,  count: 32
Region Avg IOU: 0.445448, Class: 1.000000, Obj: 0.002010, No Obj: 0.001728, Avg Recall: 0.281250,  count: 32
Region Avg IOU: 0.436241, Class: 1.000000, Obj: 0.002016, No Obj: 0.001727, Avg Recall: 0.343750,  count: 32
Region Avg IOU: 0.414918, Class: 1.000000, Obj: 0.002011, No Obj: 0.001728, Avg Recall: 0.218750,  count: 32
162: 6.710757, 11.296467 avg, 0.004000 rate, 0.154131 seconds, 20736 images
Loaded: 0.000058 seconds
Region Avg IOU: 0.470059, Class: 1.000000, Obj: 0.001975, No Obj: 0.001700, Avg Recall: 0.343750,  count: 32
```

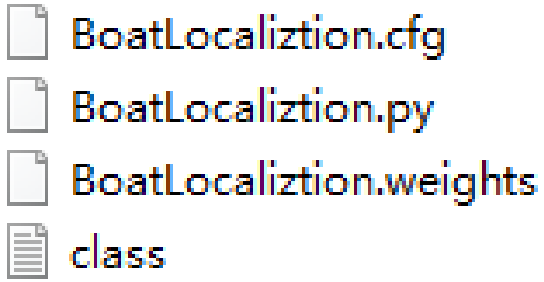
训练结束后，将得到一份**权重系数文件(weights)**

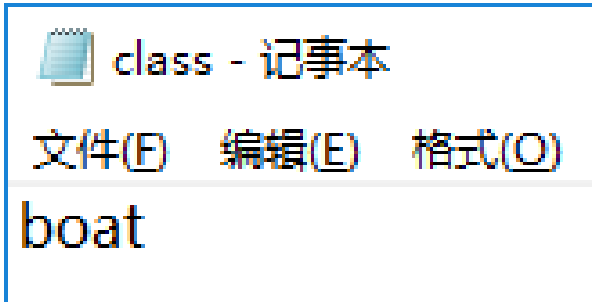
4.2 测试

测试过程跟应用过程类似，只是测试效果不好的话，需要重新训练。

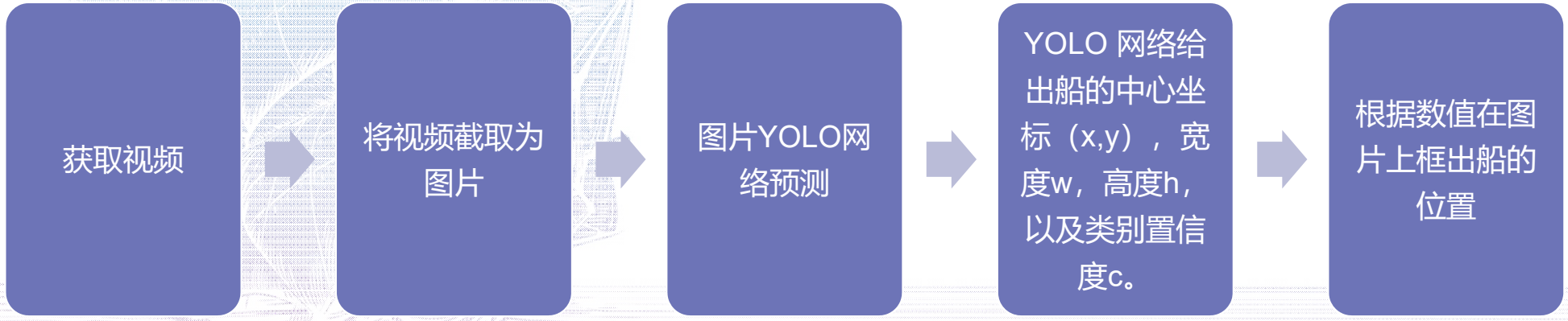
4.3 应用

应用过程需要用到几个文件：weights权重系数文件，cfg配置文件，class文件（打开，列出了物体类别，我们的应用中只有boat，如果要识别9000种物体，那就有9000个。），执行应用的python文件。





执行应用的流程包括：



4.3 应用

执行程序。（程序代码展示）

写好执行程序后，要将几个文件放置于同一个文件夹中，然后运行执行程序，即可开始无人船定位识别。（演示）