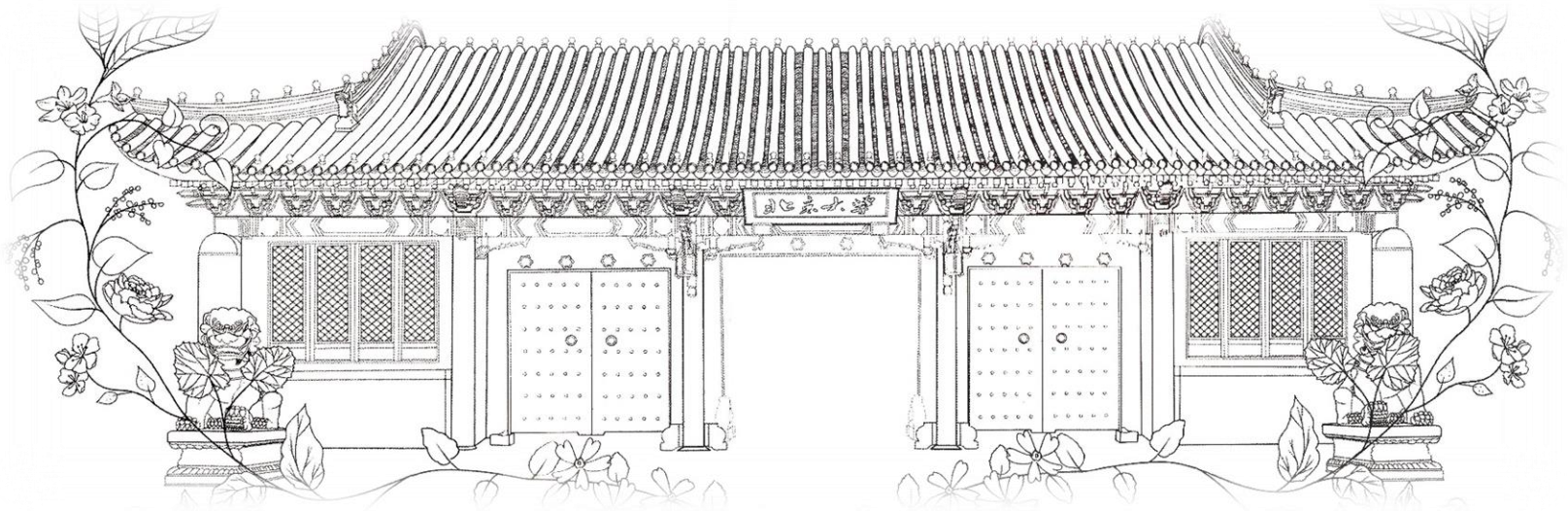
# 《Python数据分析》
# 图像数据处理与分析

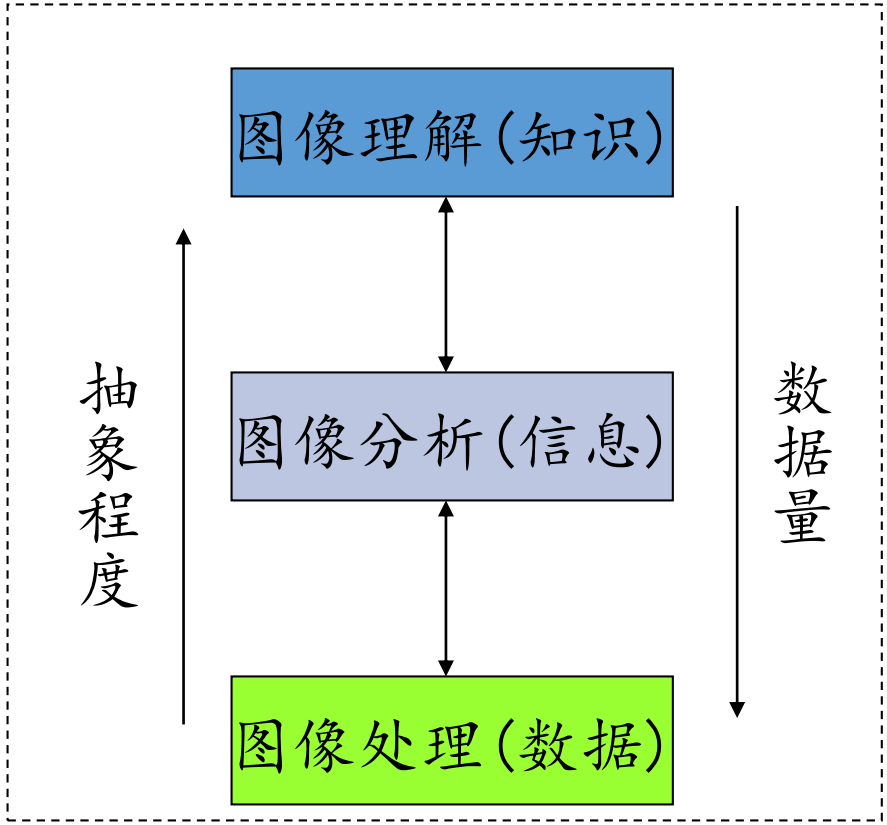


# 目录

- 图像基础知识
- 图像数据分析应用场景
- 图像数据基本操作
- 高级任务

# 图像基础知识

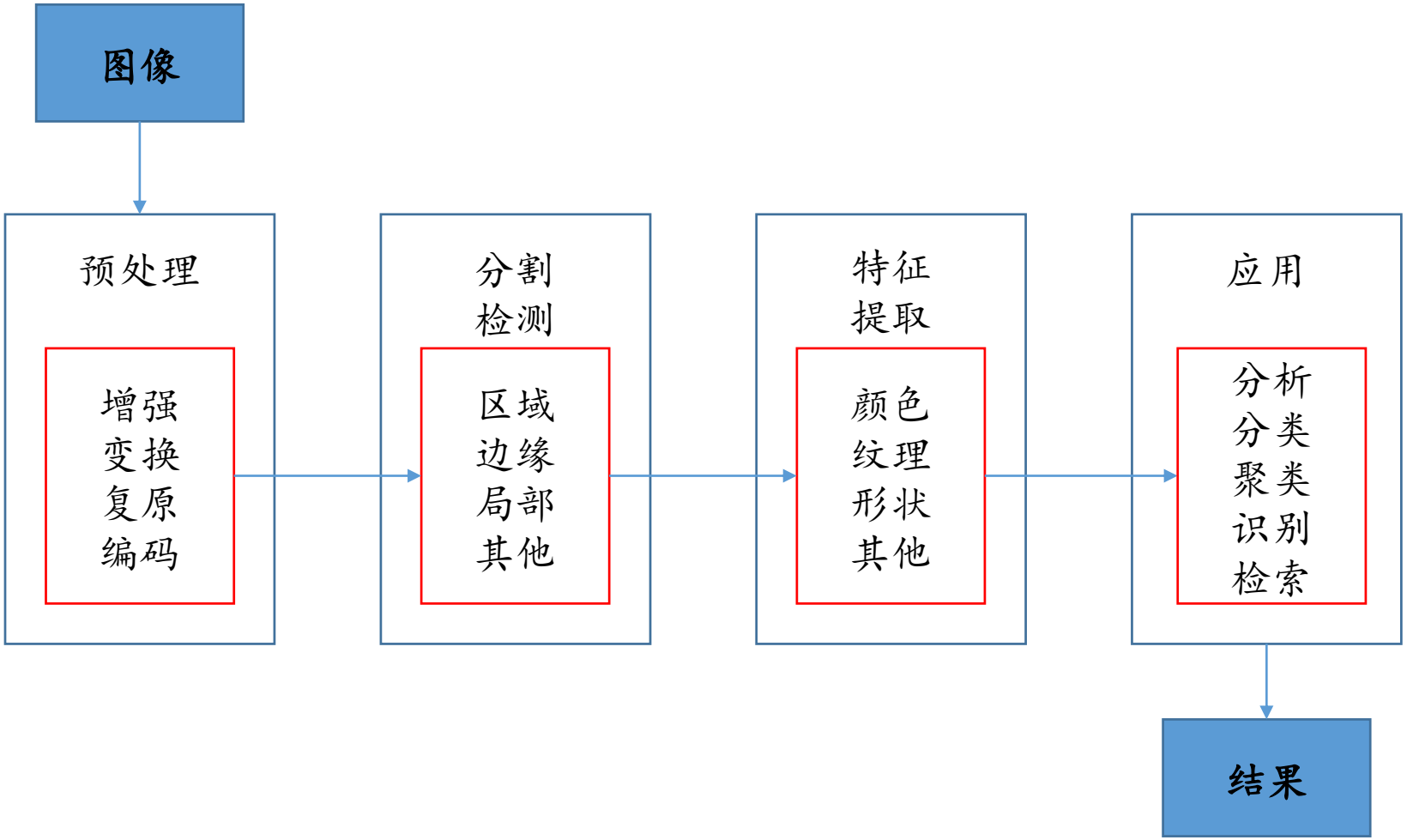
- 图像(image)是泛指照片、动画等等形成视觉景象的事物。
- 图像与计算机图形学中的图形的区别是
  - 计算机图形学是从建立数学模型到生成图形
  - 图像通常是指从外界产生的图形。



# 图像处理技术

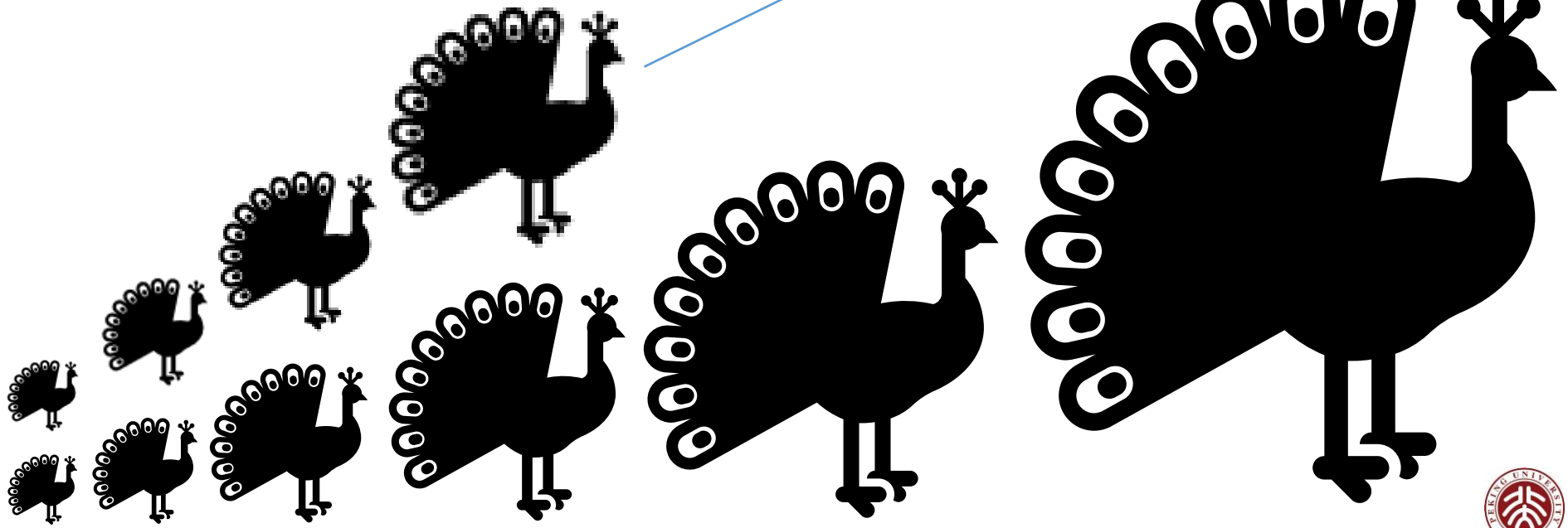
- 图像处理技术是与图像有关的技术总称。
  - 图像的形成、存储和显示
    - 采集、编码、存储和传输、生成、增强、恢复和重建
  - 图像的处理
    - 分割、变换、检测、描述、特征提取、分类、识别。
- 狭义的数字图像处理是指图像的增强、复原和重建，操作的对像是图像的像素，输出的是图像。

# 图像处理技术路径


图像
预处理
增强
变换
复原
编码
分割
检测
区域
边缘
局部
其他
特征
提取
颜色
纹理
形状
其他
应用
分析
分类
聚类
识别
检索
结果

# 图像的种类

- 矢量图(vector graphics)
- 位图(bitmap，bitmapped image )
  - 灰度图(gray-scale image)
  - 彩色图像(color image)



# 图像的种类

- 灰度图(gray-scale image)
- 彩色图像(color image)
  - 常见格式:

  BMP（BitMaP）格式

  JPEG（Joint Photographic Expert Group）格式（JPG）

  PNG（Portable Network Graphics ）格式

# 位图(bitmap，bitmapped image )

- 用像素值阵列表示的图
  - 对位图进行操作时，只能**对图中的像素进行操作**，而不能把位图中的物体作为独立实体进行操作。
  - 存储的内容是描述像素的数值
- 特性
  - 位图的获取通常用扫描仪、相机、摄像机、录像机、光盘和相关的数字化设备
  - 位图文件占据的存储空间比较大
  - 影响位图文件大小的因素
    - 图像分辨率：分辨率越高，表示组成一幅图的像素就越多，图像文件就越大
    - 像素深度：像素深度越深，表达单个像素的颜色和亮度的位数越多，图像文件就越大

# 图像分辨率

- 在图像显示应用中的图像分辨率表示法
  - (1) 物理尺寸：每毫米线数(或行数)
  - (2) 行列像素：像素/行×像素/列，
    如640像素/行×480像素/列
  - (3) 像素总数：如数码相机上标的500万像素
  - (4) 单位长度上的像素：如像素每英寸(pixels per inch, PPI)
  - (5) 线对(line pair)数：以黑白相邻的两条线为一对，如5对线

# 位图图像表示

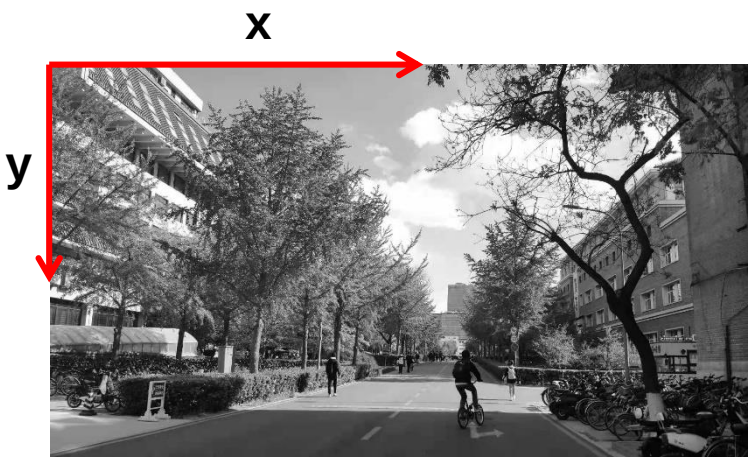
- 灰度图像是一个二维灰度（或亮度）函数f(x,y)
- 彩色图像由三个（如RGB,）二维值函数f(x,y)组成
- 在skimage中，都是np.ndarray类型

Shape： (1080, 1920, 3)

Shape： (1080, 1920,)

# 位图图像表示

灰度图像




```
(road_image_grey*255).astype('uint8')[:10,:10]
```

```
array([[192, 101, 112, 104, 148, 203, 130, 131, 131, 121],
       [121,  76, 103, 113, 116, 132, 129, 154, 137, 126],
       [120,  90, 137, 162, 139, 136, 135, 142, 139, 131],
       [119, 110, 143, 154, 150, 184, 182, 130, 136, 129],
       [115, 103,  96,  98, 109, 154, 181, 146, 128, 124],
       [120, 121, 103, 119, 114,  89, 112, 133, 121, 120],
       [121, 168, 146, 159, 157,  88,  81, 109, 117, 119],
       [127, 192, 146, 129, 159, 125, 112, 116, 118, 121],
       [131, 207, 166, 117, 172, 169, 109, 113, 126, 126],
       [134, 185, 151, 114, 152, 155, 123, 134, 121, 119]], dtype=uint8)
```

或者，

```
road_image_grey[:10,:5]
```

```
array([[0.52123412, 0.39998627, 0.44303451, 0.40966863, 0.50050196],
       [0.47453137, 0.30020549, 0.40605882, 0.44496275, 0.45501686],
       [0.47332706, 0.35482471, 0.53910941, 0.63543804, 0.54521294],
       [0.47051451, 0.43493059, 0.56431333, 0.60742157, 0.59170627],
       [0.45229843, 0.4049498 , 0.37803529, 0.38584941, 0.43092196],
       [0.47329804, 0.47664706, 0.40435529, 0.46763686, 0.44996431],
       [0.47664706, 0.66178706, 0.57437451, 0.62617412, 0.61719294],
       [0.42357137, 0.7556149 , 0.57548353, 0.50767882, 0.62447059],
       [0.51611647, 0.81190863, 0.6513851 , 0.46229451, 0.67545098],
       [0.52842471, 0.72646039, 0.59395333, 0.44856549, 0.59645412]])
```

# 位图图像表示

彩色图像


图像尺寸： 1920 * 1080

X * Y

```
road_image.transpose((2, 0, 1))[:, :10, :10]
```

```
array([[[135, 104, 114, 104, 140, 201, 127, 129, 130, 121],
        [124,  78, 104, 113, 114, 130, 126, 152, 135, 126],
        [123,  92, 138, 161, 137, 134, 133, 140, 138, 130],
        [122, 112, 144, 154, 149, 182, 180, 128, 135, 129],
        [117, 104,  96,  97, 108, 152, 179, 144, 127, 124],
        [122, 122, 103, 118, 113,  86, 110, 132, 120, 120],
        [122, 169, 146, 158, 155,  86,  79, 108, 117, 120],
        [118, 192, 146, 128, 157, 123, 110, 115, 118, 122],
        [131, 206, 165, 116, 170, 167, 108, 112, 126, 127],
        [134, 184, 150, 112, 150, 153, 121, 134, 122, 121]],

       [[134, 104, 117, 111, 157, 214, 142, 142, 141, 129],
        [123,  79, 108, 120, 125, 143, 140, 165, 146, 134],
        [122,  93, 142, 169, 148, 147, 146, 152, 148, 138],
        [122, 114, 149, 162, 160, 195, 193, 140, 145, 136],
        [118, 107, 102, 106, 119, 165, 192, 156, 136, 131],
        [124, 126, 109, 127, 124,  99, 122, 143, 129, 126],
        [126, 174, 153, 168, 167,  98,  91, 118, 125, 125],
        [128, 199, 154, 138, 169, 135, 122, 125, 125, 126],
        [138, 214, 174, 127, 182, 179, 119, 121, 132, 130],
        [142, 193, 160, 124, 162, 165, 132, 142, 127, 123]],

       [[116,  76,  70,  41,  65, 103,  36,  36,  47,  54],
        [103,  48,  58,  50,  33,  37,  29,  60,  54,  59],
        [101,  61,  91,  96,  56,  42,  38,  50,  59,  65],
        [ 94,  77,  93,  87,  66,  90,  87,  40,  58,  68],
        [ 84,  64,  42,  27,  25,  60,  87,  58,  53,  64],
        [ 84,  76,  45,  46,  28,   0,  20,  49,  50,  64],
        [ 76, 116,  83,  82,  69,   0,   0,  29,  50,  69],
        [ 76, 132,  77,  49,  69,  33,  24,  39,  57,  75],
        [ 70, 141,  91,  33,  82,  77,  25,  40,  68,  85],
        [ 65, 112,  71,  26,  60,  65,  40,  65,  69,  83]]], dtype=uint8)
```

R(红) G(绿) B(蓝)

# 位图图像表示

- 灰度图像是一个二维灰度（或亮度）函数f(x,y)
- 彩色图像由三个（如RGB,）二维值函数f(x,y)组成
- 在skimage中，都是np.ndarray类型

Shape： (1080, 1920, 3)

Shape： (1080, 1920,)

# 目录

- 图像基础知识
- 图像处理与分析应用场景
- 图像数据基本操作
- 高级任务

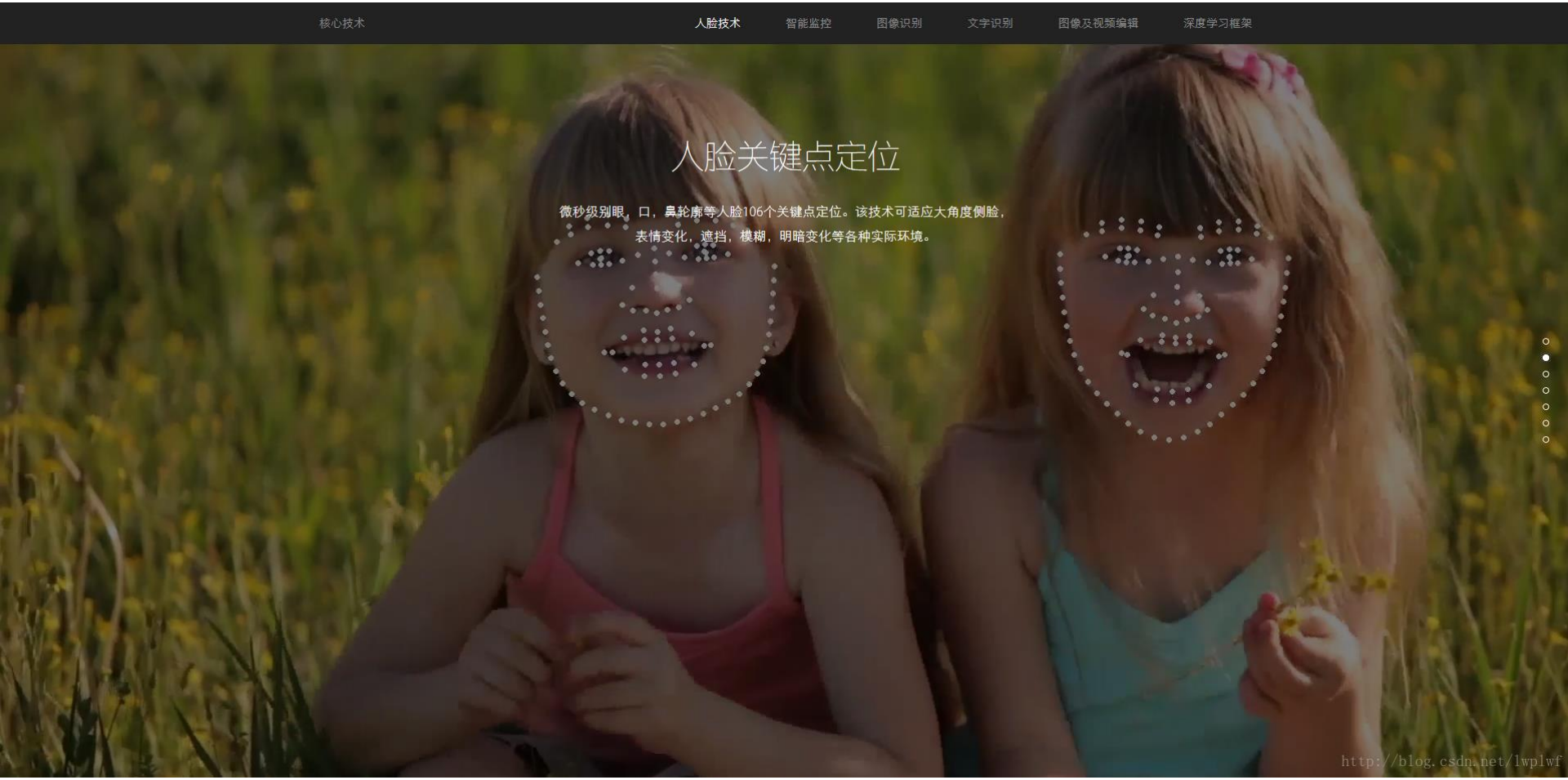
# 图像数据分析应用场景

- 图像变换
- 图像编码压缩
- 图像增强和复原
- 图像分割
- 图像描述
- 图像分类（识别）

# 人脸检测跟踪


核心技术
人脸技术
智能监控
图像识别
文字识别
图像及视频编辑
深度学习框架
人脸检测跟踪
对背景复杂低质量的图片或百人人群监控视频，可以在移动设备和个人电脑上实现毫秒级别的人脸检测。
该技术可适应侧脸，遮挡，模糊，表情变化等各种实际环境。
4
5
1
16
17
3
20
21
http://blog.csdn.net/lwplwf

# 人脸关键点检测



# 人像美化

核心技术 人脸技术 智能监控 图像识别 文字识别 图像及视频编辑 深度学习框架

## 人像美颜/美妆

基于智能人脸检测定位技术，打造移动端美颜、美妆效果解决方案，
让移动互联网娱乐时代有“美”可依。

美颜 美妆

原图 瘦脸 磨皮 美白 红润





# 人像美化

B612 & FaceU……

# 人像美化

B612 & FaceU……

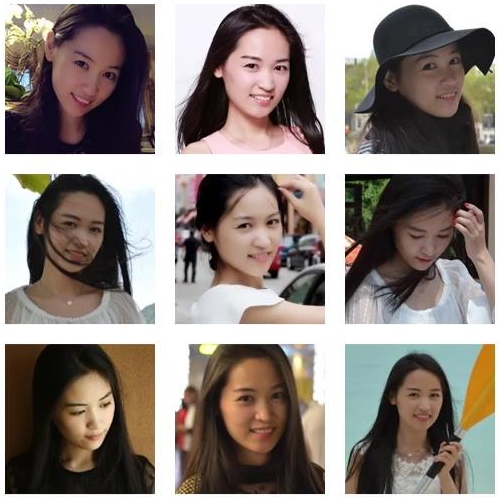

# 人脸聚类



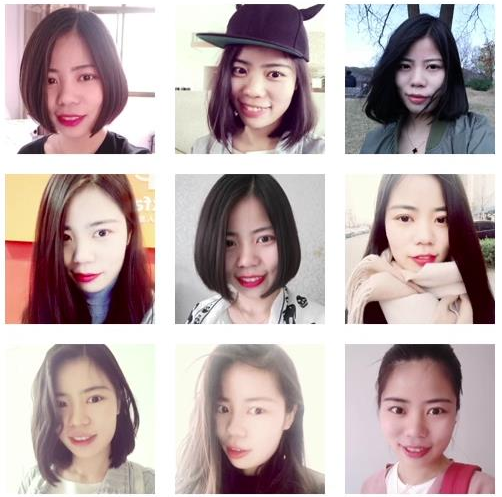
## 人脸聚类

数十万人的人脸快速聚类，可用于基于人脸的智能相册以及基于合影的社交网络分析。让照片管理更直观，让社交关系更清晰。

Elva



Ramy



Coral

# 真人检测

核心技术　人脸技术　智能监控　图像识别　文字识别　图像及视频编辑　深度学习框架

## 真人检测

检测摄像头前用户是否为真人操作，配合人脸身份认证，为金融等高安全性要求的严肃应用场景提供真人身份验证。
能有效分辨高清照片，PS，三维模型，换脸等仿冒欺诈。我们为用户配合和用户不配合场景提供解决方案。

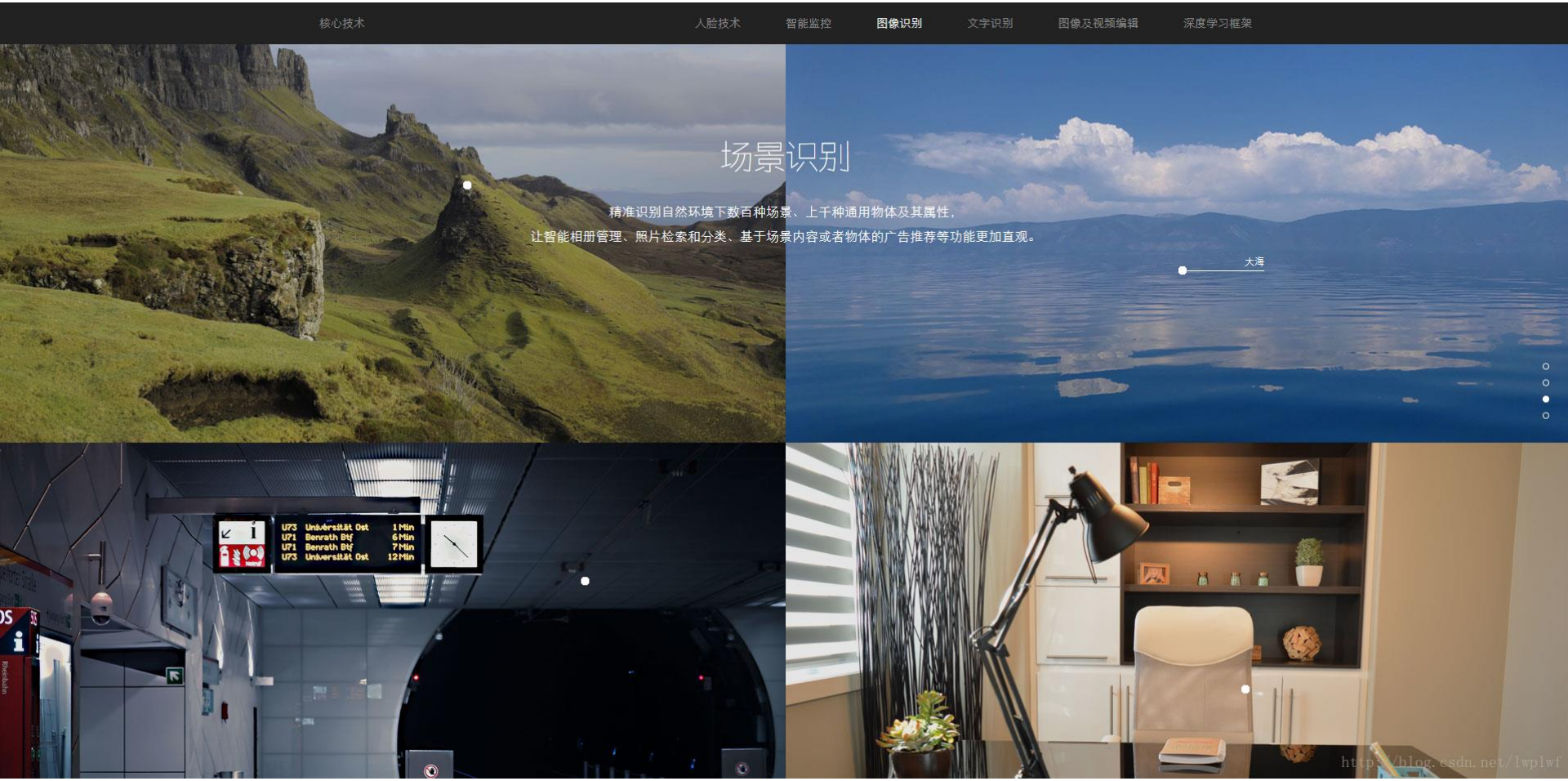
真人验证
liveness Detection

请眨眼
Blink

仿冒验证
Fake Liveness Detection

# 物体检测

# 场景识别



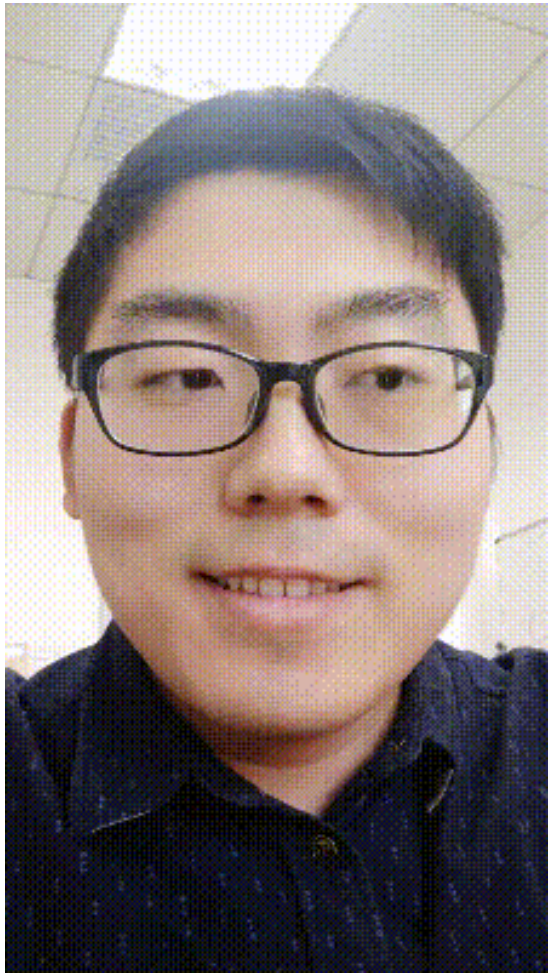
# 智能相机

## AI 场景相机
## 不可思议的 AI 梦境虚化效果

拥有一部超能相机是什么感觉？为人像照片带来梦境般的动态背景虚化效果，智能识别 206 种拍照场景，实时优化调校。甚至拍 PPT 都能超分辨率拍摄，内容清晰锐利，神奇之处待你体验。

# 照片上色

# 表情包制作



# 风格迁移（Style Transfer）

Content

Style

# 风格迁移（Style Transfer）

# 目录

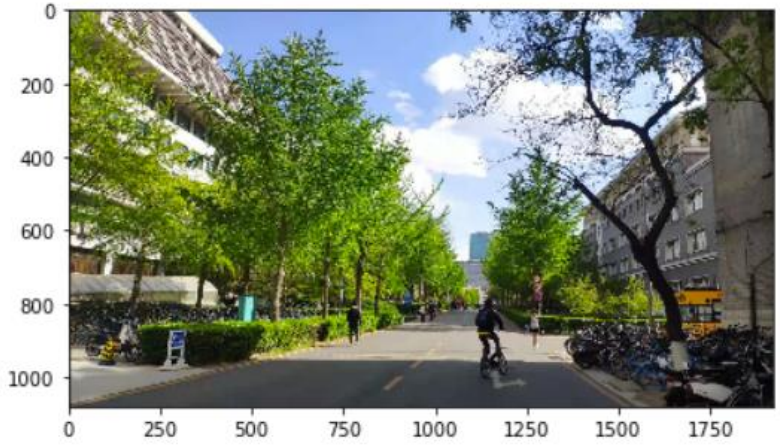
- 图像基础知识
- 图像处理与分析应用场景
- 图像数据基本操作
- 图像统计
- 高级任务

# 图像读写

主要利用**skimage.io**模块

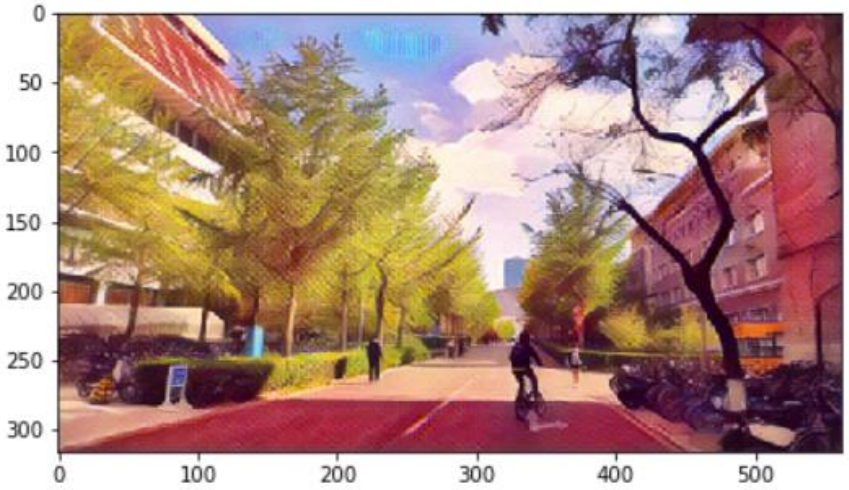
```
from skimage import io
road_image = io.imread("road.jpg")
io.imshow(road_image)
```

```
<matplotlib.image.AxesImage at 0x235c338a6d0>
```




```
road_candy = io.imread("road_candy.png")
io.imshow(road_candy)
```

```
<matplotlib.image.AxesImage at 0x22c9386d130>
```



# 图像读写

```
io.imsave("road2.jpg",road_image)
```

```
io.imsave("road_candy2.png",road_candy)
```

road_candy2.png

road2.jpg

```
io.imsave("road3.png",road_image)
```

```
io.imsave("road_candy3.jpg",road_candy)
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-25-ad490273f36f> in <module>
----> 1 io.imsave("road_candy3.jpg",road_candy)

……

OSError: JPEG does not support alpha channel.
```

JPG： W*H*3

PNG： W*H*4

# 图像切片

- 读入的图像数据以np.ndarray格式存储，数据类型为uint8，取值为0~255的整数。可以用处理ndarray的方式进行数据切片和访问。

```
type(road_image)
```

```
numpy.ndarray
```

```
road_image.dtype
```

```
dtype('uint8')
```

```
road_image.shape
```

```
(1080, 1920, 3)
```

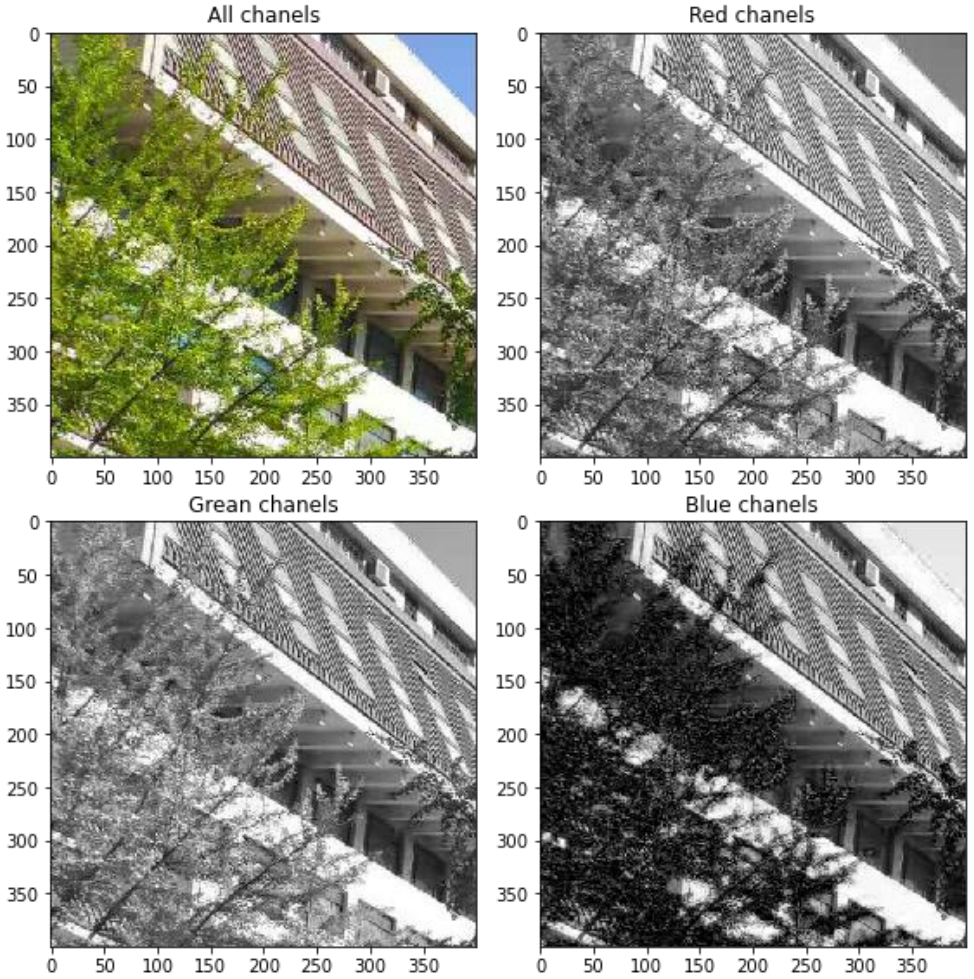
```
road_image[:400,:400,0]
```

```
array([[135, 104, 114, ..., 123, 123, 123],
       [124,  78, 104, ..., 123, 123, 123],
       [123,  92, 138, ..., 123, 123, 123],
       ...,
       [209, 175, 135, ..., 178,  93,  60],
       [208, 229, 218, ..., 146,  83,  18],
       [190, 196, 187, ..., 171, 130,  72]], dtype=uint8)
```

# 图像切片

- 切片可以访问不同颜色通道和区域：

```
part_of_road = road_image[:400,:400,:]
plt.figure(figsize=(8,8))
plt.subplot(2,2,1)
io.imshow(part_of_road)
plt.title("All chanels")
plt.subplot(2,2,2)
io.imshow(part_of_road[:,:,0])
plt.title("Red chanels")
plt.subplot(2,2,3)
io.imshow(part_of_road[:,:,1])
plt.title("Grean chanels")
plt.subplot(2,2,4)
io.imshow(part_of_road[:,:,2])
plt.title("Blue chanels")
plt.show()
```



# 改变图像大小

- 利用skimage.transform.resize，指定resize后的尺寸

```
road_image.shape
```

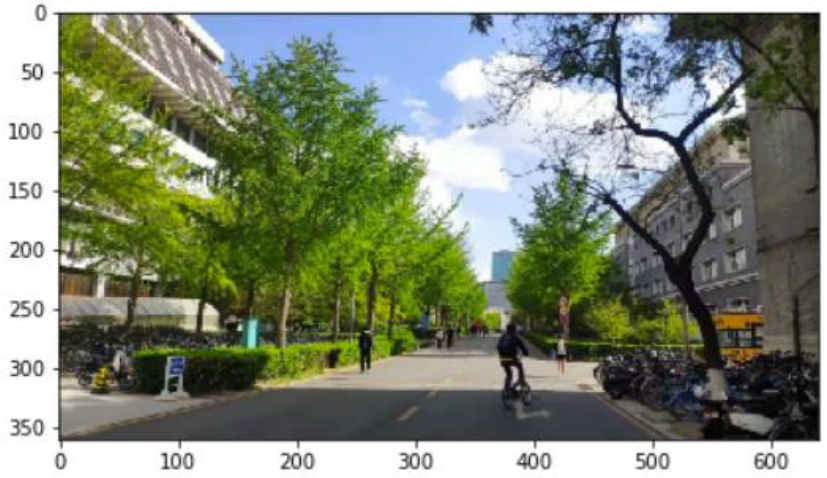
```
(1080, 1920, 3)
```

```
road_image_resized = transform.resize(road_image,(360,640))
road_image_resized.shape
```

```
(360, 640, 3)
```

```
io.imshow(road_image_resized)
```

```
<matplotlib.image.AxesImage at 0x235d5b2c0d0>
```

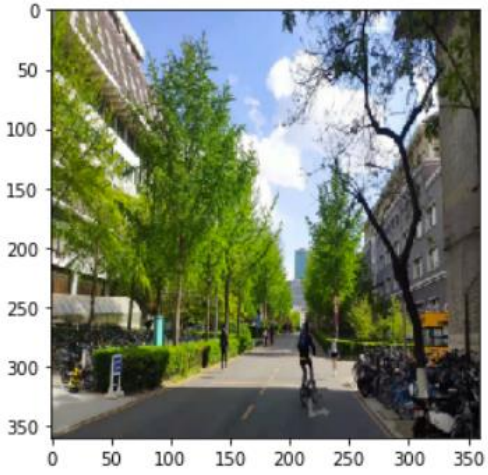



```
road_image_resized2 = transform.resize(road_image,(360,360))
road_image_resized2.shape
```

```
(360, 360, 3)
```

```
io.imshow(road_image_resized2)
```

```
<matplotlib.image.AxesImage at 0x235d60778b0>
```



# 改变图像大小

- 利用skimage.transform.rescale，指定缩放比例

```
road_image.shape
```

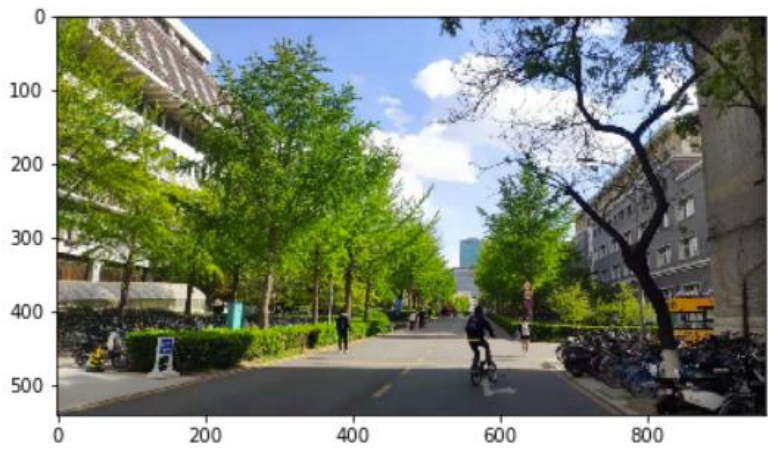
```
(1080, 1920, 3)
```

```
road_image_rescaled = transform.rescale(road_image, (0.5, 0.5, 1))
road_image_rescaled.shape
```

```
(540, 960, 3)
```

```
io.imshow(road_image_rescaled)
```

```
<matplotlib.image.AxesImage at 0x235d56b7f70>
```



# 改变图像大小

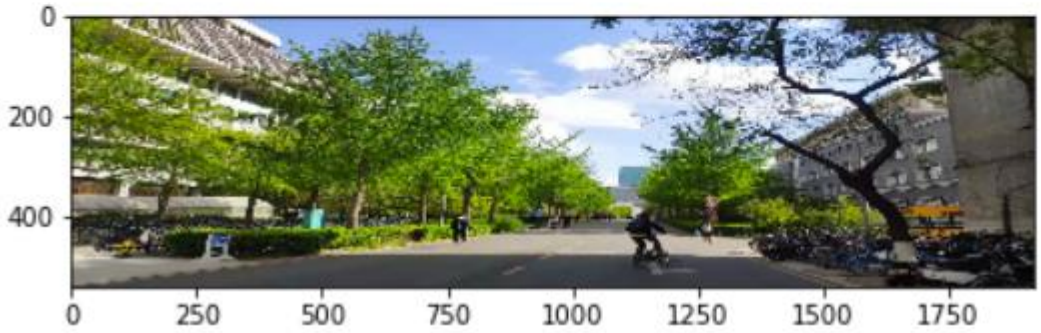
- 利用skimage.transform.rescale，指定缩放比例

```
road_image_rescaled2 = transform.rescale(road_image,(0.5,1,1))
road_image_rescaled2.shape
```

```
(540, 1920, 3)
```

```
io.imshow(road_image_rescaled2)
```

```
<matplotlib.image.AxesImage at 0x235d596fb80>
```



# 改变图像大小

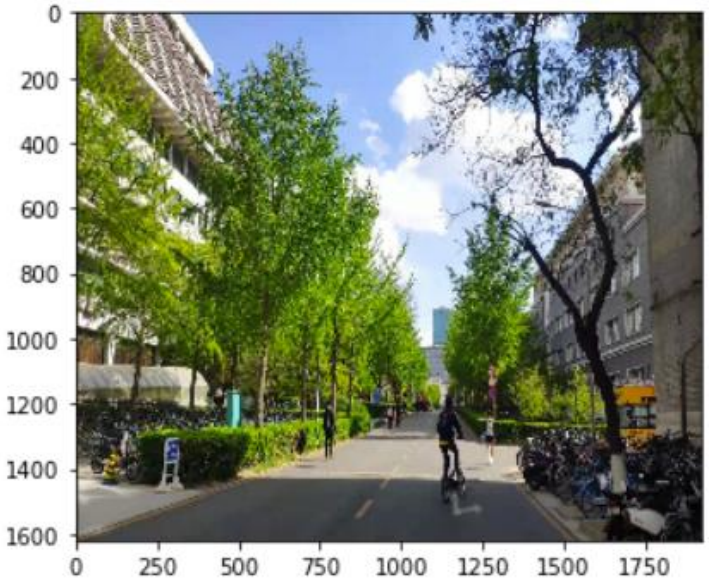
- 利用skimage.transform.rescale，指定缩放比例

```
road_image_rescaled3 = transform.rescale(road_image,(1.5,1,1))
road_image_rescaled3.shape
```

```
(1620, 1920, 3)
```

```
io.imshow(road_image_rescaled3)
```

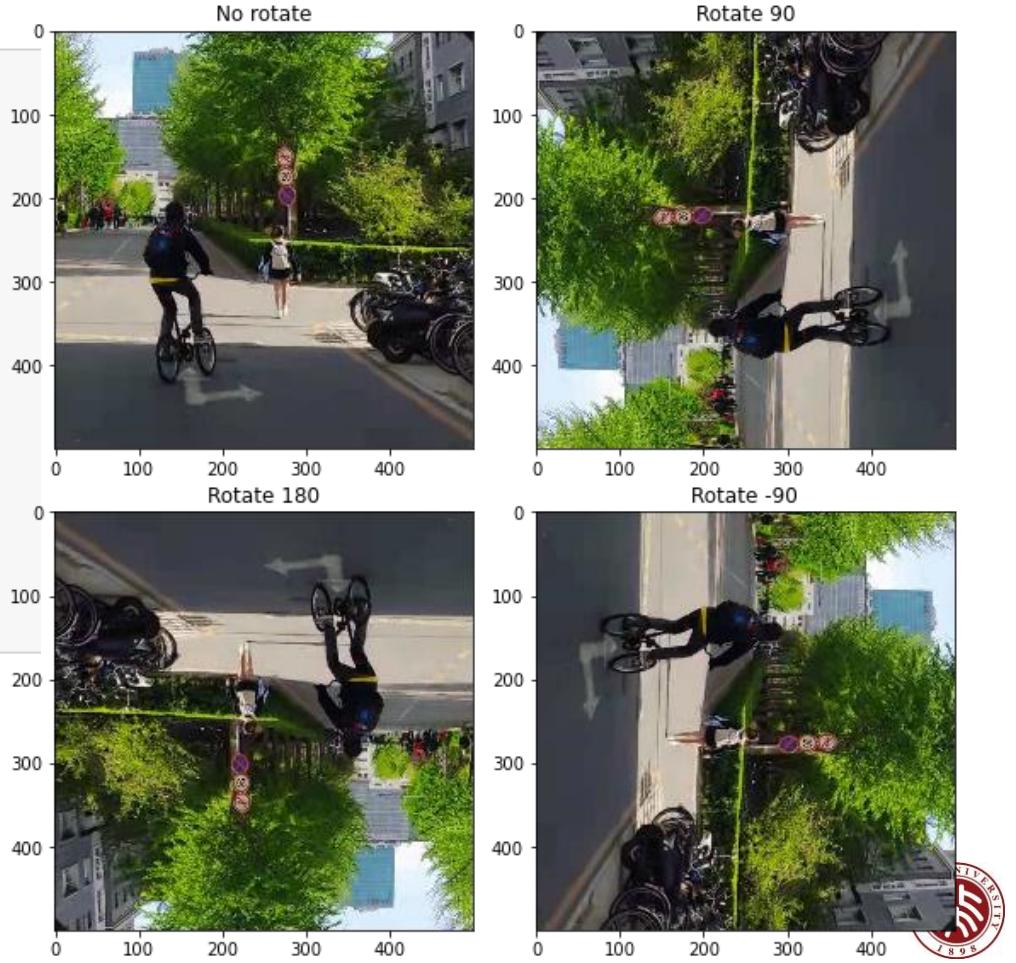
```
<matplotlib.image.AxesImage at 0x235d5a55040>
```



# 图像旋转

- 利用skimage.transform.rotate，指定转换角度

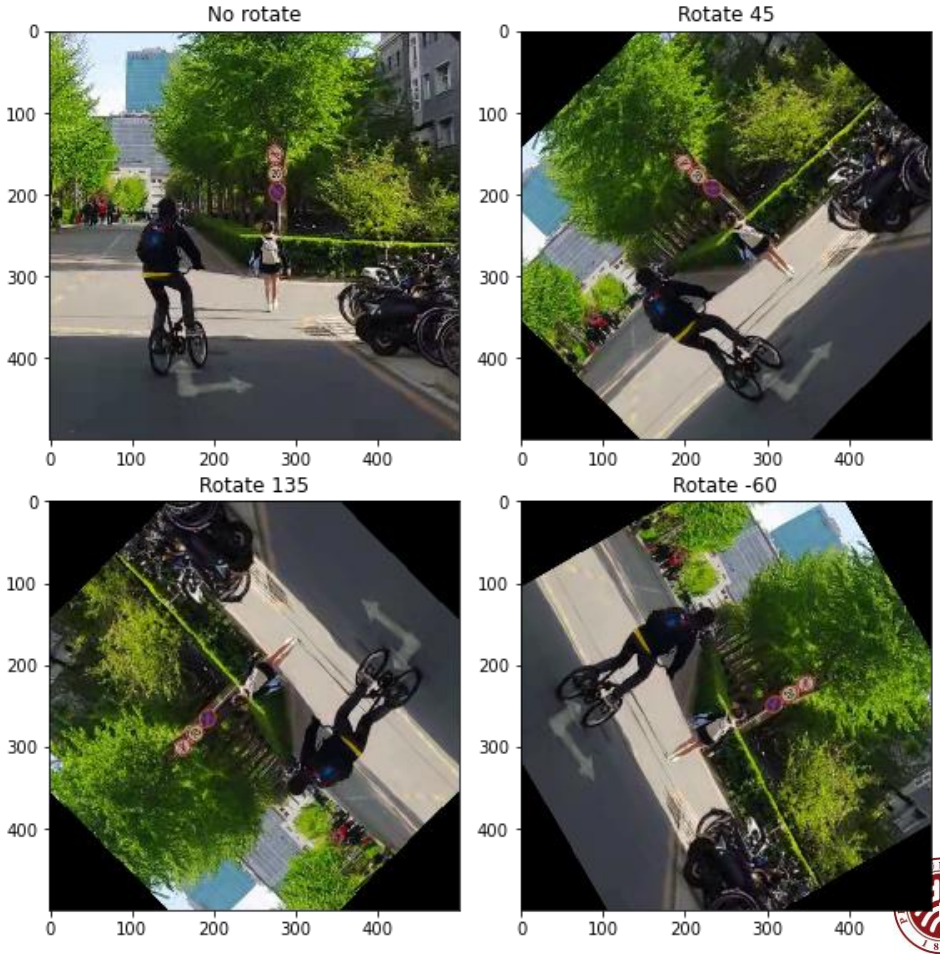
```
person_in_road = road_image[-500:,1000:1500,:]
plt.figure(figsize=(8,8))
plt.subplot(2,2,1)
io.imshow(person_in_road)
plt.title("No rotate")
plt.subplot(2,2,2)
io.imshow(transform.rotate(person_in_road,90))
plt.title("Rotate 90")
plt.subplot(2,2,3)
io.imshow(transform.rotate(person_in_road,180))
plt.title("Rotate 180")
plt.subplot(2,2,4)
io.imshow(transform.rotate(person_in_road,-90))
plt.title("Rotate -90")
plt.show()
```



# 图像旋转

- 利用skimage.transform.rotate，指定转换角度

```
person_in_road = road_image[-500:,1000:1500,:]
plt.figure(figsize=(8,8))
plt.subplot(2,2,1)
io.imshow(person_in_road)
plt.title("No rotate")
plt.subplot(2,2,2)
io.imshow(transform.rotate(person_in_road,45))
plt.title("Rotate 45")
plt.subplot(2,2,3)
io.imshow(transform.rotate(person_in_road,135))
plt.title("Rotate 135")
plt.subplot(2,2,4)
io.imshow(transform.rotate(person_in_road,-60))
plt.title("Rotate -60")
plt.show()
```



# 图像对比度

- 利用skimage.exposure. is_low_contrast判读是否低对比度，利用skimage.exposure.adjust_gamma进行伽马调整

```
from skimage import exposure
```

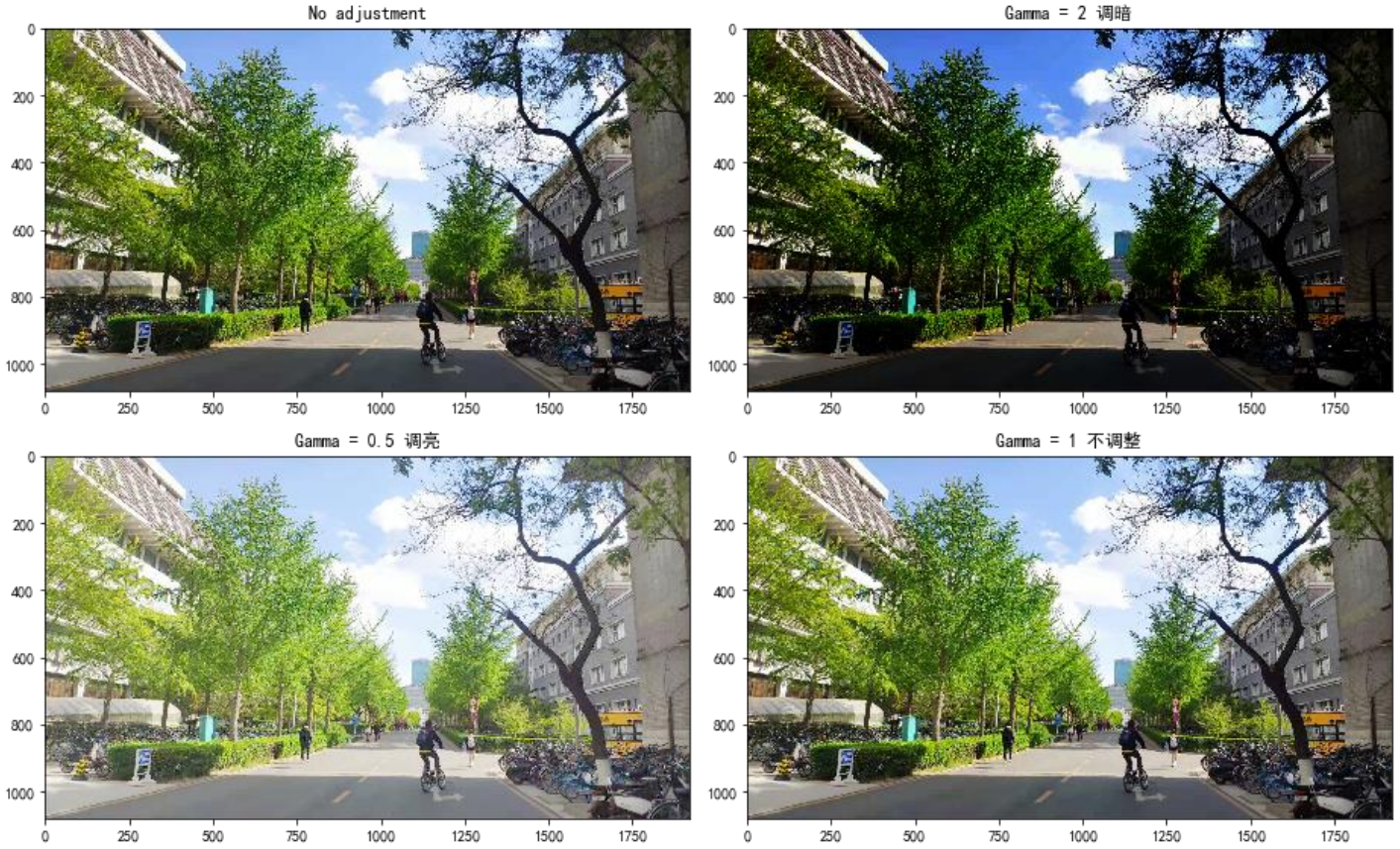
```
exposure.is_low_contrast(road_image)
```

```
False
```

```
plt.figure(figsize=(13,8))
plt.subplot(2,2,1)
io.imshow(road_image)
plt.title("No adjustment")
plt.subplot(2,2,2)
io.imshow(exposure.adjust_gamma(road_image, 2)) # >1 调暗
plt.title("Gamma = 2 调暗")
plt.subplot(2,2,3)
io.imshow(exposure.adjust_gamma(road_image, 0.5)) # <1 调亮
plt.title("Gamma = 0.5 调亮")
plt.subplot(2,2,4)
io.imshow(exposure.adjust_log(road_image, 1))
plt.title("Gamma = 1 不调整")
plt.show()
```

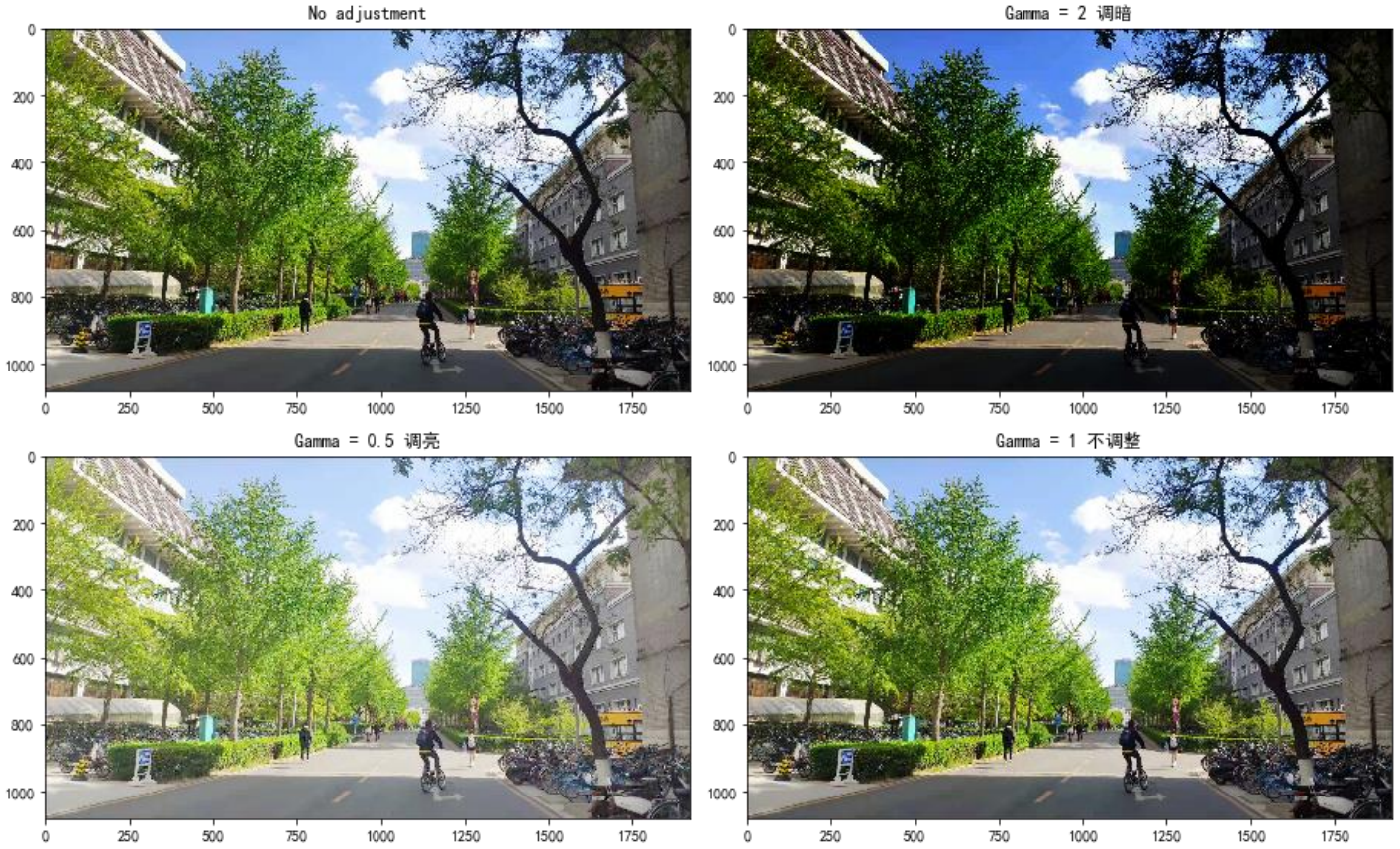
# 图像对比度

- 利用skimage.exposure.adjust_gamma进行伽马调整



# 图像对比度

- 利用skimage.exposure.adjust_gamma进行伽马调整




skimage.exposure.adjust_log也可进行对比度调整

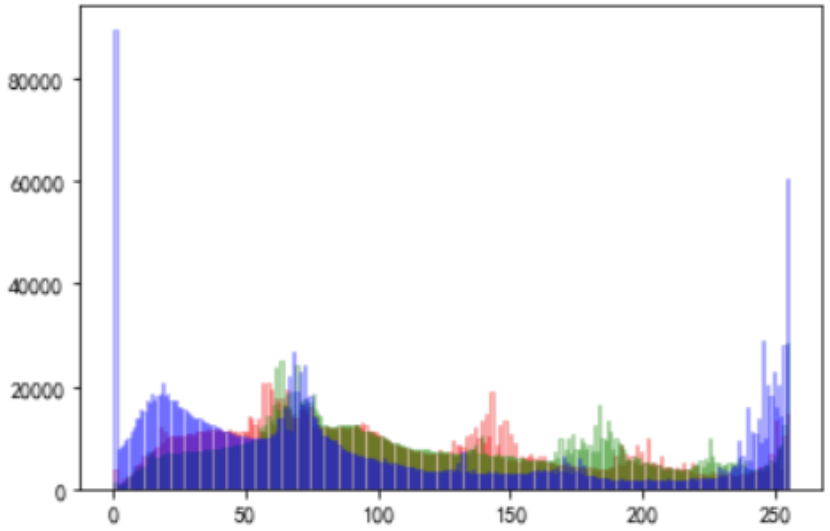
# 目录

- 图像基础知识
- 图像处理与分析应用场景
- 图像数据基本操作
- 高级任务

# 图像直方图

- 绘制颜色直方图

```
red = road_image[:,:,0].flatten()
plt.hist(red, bins=256, facecolor='r',alpha=0.25, edgecolor='r',)
green = road_image[:,:,1].flatten()
plt.hist(green, bins=256, facecolor='g',alpha=0.25, edgecolor='g')
blue = road_image[:,:,2].flatten()
plt.hist(blue, bins=256, facecolor='b',alpha=0.25, edgecolor='b')
plt.show()
```



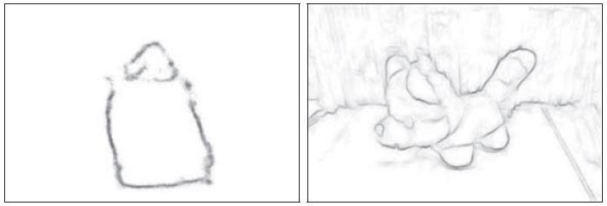
# 目录

- 图像基础知识
- 图像处理与分析应用场景
- 图像数据基本操作
- 高级任务

# 边缘检测

- 物体的边缘是以图像**局部特性的不连续性**的形式出现的。边缘意味着一个区域的终结和另一个区域的开始。
- 图像边缘信息在图像分析和人的视觉中都是十分重要的，是图像识别中提取图像特征的一个重要属性。

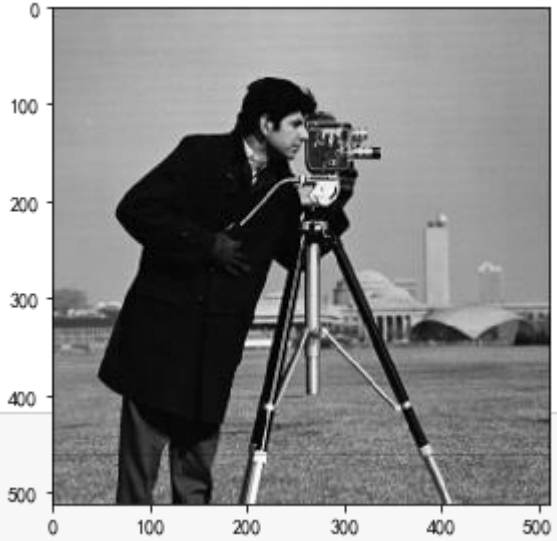
(a) A picture of mouse　(b) A picture of toy



(c) Predicted mouse edge　(d) Predicted toy edge

# 边缘检测

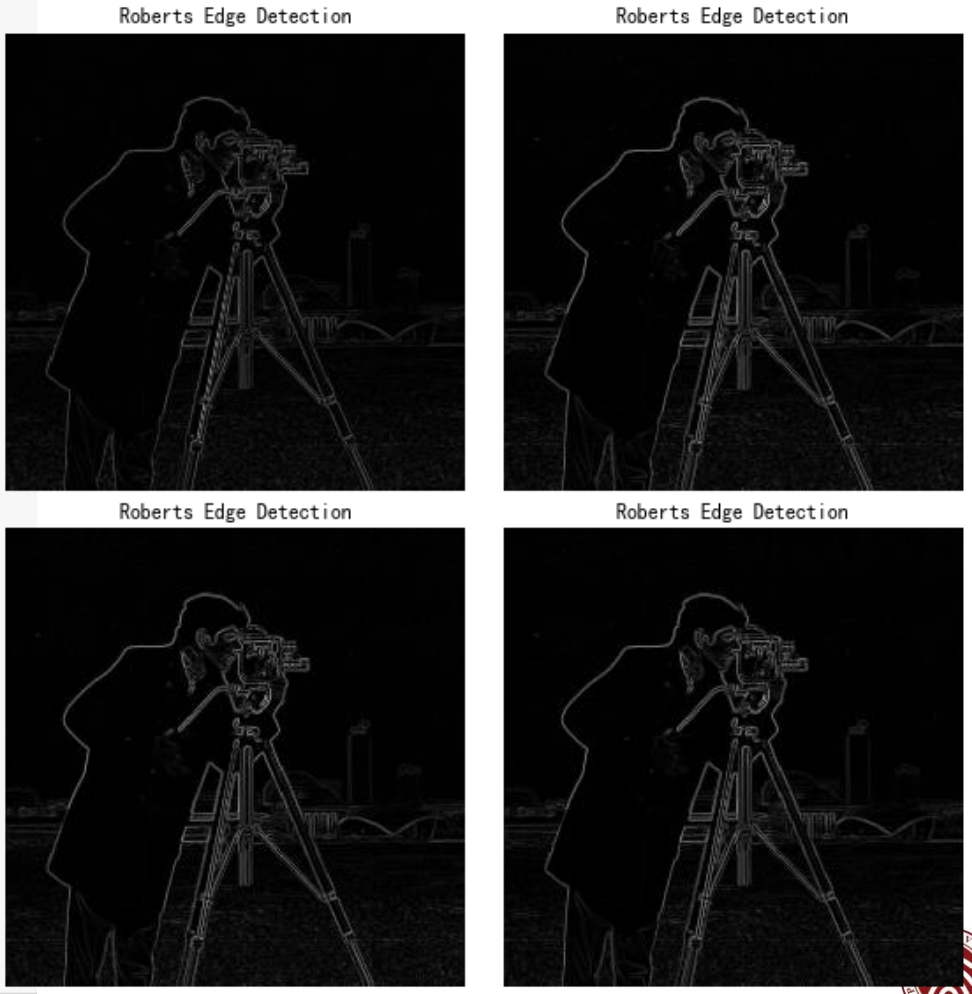
- 利用skimage.filters和skimage.features进行边缘检测
  - 常用算子：
  - Roberts,
  - Sobel,
  - Scharr,
  - Prewitt
  - Canny




```
import numpy as np
import matplotlib.pyplot as plt

from skimage.data import camera
from skimage.filters import roberts, sobel, scharr, prewitt
from skimage.color import rgb2gray

image = camera()
edge_roberts = roberts(image)
edge_sobel = sobel(image)
edge_scharr = scharr(image)
edge_prewitt = prewitt(image)
```

# 边缘检测

```
plt.subplots(nrows =2, ncols=2, figsize=(8, 8))

plt.subplot(2,2,1)
io.imshow(edge_roberts, cmap=plt.cm.gray)
plt.title('Roberts Edge Detection')
plt.axis('off')

plt.subplot(2,2,2)
io.imshow(edge_sobel, cmap=plt.cm.gray)
plt.title('Roberts Edge Detection')
plt.axis('off')

plt.subplot(2,2,3)
io.imshow(edge_scharr, cmap=plt.cm.gray)
plt.title('Roberts Edge Detection')
plt.axis('off')

plt.subplot(2,2,4)
io.imshow(edge_prewitt, cmap=plt.cm.gray)
plt.title('Roberts Edge Detection')
plt.axis('off')

plt.tight_layout()
plt.show()
```

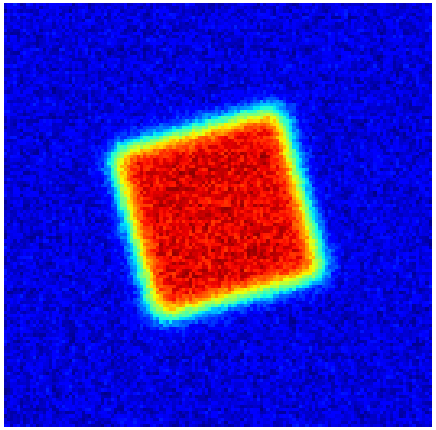


# Canny算子

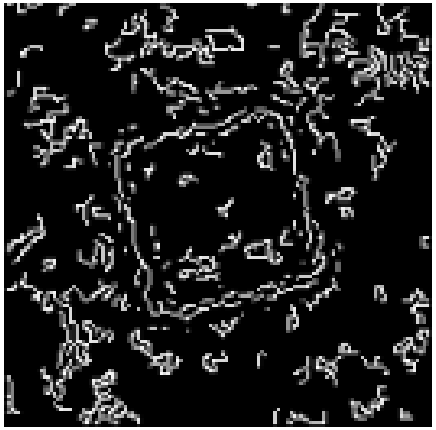
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage as ndi
from skimage import feature

# Generate noisy image of a square
im = np.zeros((128, 128))
im[32:-32, 32:-32] = 1

im = ndi.rotate(im, 15, mode='constant')
im = ndi.gaussian_filter(im, 4)
im += 0.2 * np.random.random(im.shape)

# Compute the Canny filter for two values of sigma
edges1 = feature.canny(im)
edges2 = feature.canny(im, sigma=3)

# display results
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(8, 3), sharex=True, sharey=True)

ax1.imshow(im, cmap=plt.cm.jet)
ax1.axis('off')
ax1.set_title('noisy image', fontsize=12)

ax2.imshow(edges1, cmap=plt.cm.gray)
ax2.axis('off')
ax2.set_title('Canny filter, $\sigma=1$', fontsize=12)

ax3.imshow(edges2, cmap=plt.cm.gray)
ax3.axis('off')
ax3.set_title('Canny filter, $\sigma=3$', fontsize=12)

plt.show()
```

# Canny算子

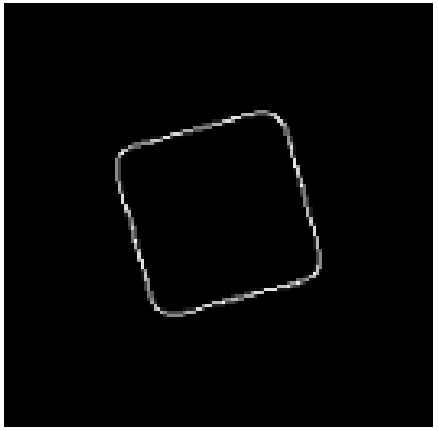
noisy image



Canny filter, $\sigma = 1$

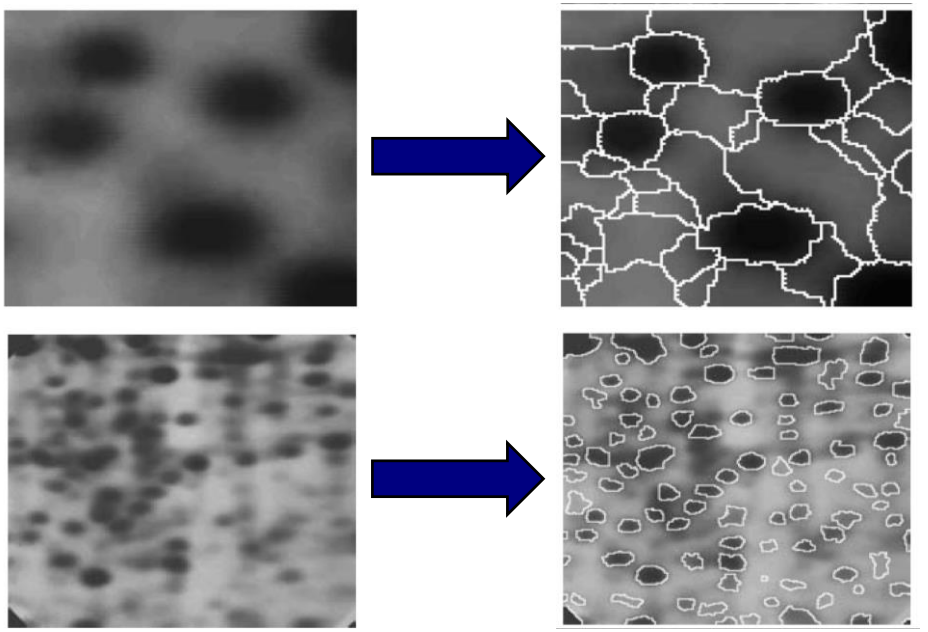


Canny filter, $\sigma = 3$



# 图像分割

- 根据图像中的物体将图像的像素分类，并提取感兴趣目标
- 图像分割是图像识别和图像理解的基本前提步骤

# 图像识别

- 识别图像一般指对图像进行处理、分析和理解，以识别各种不同模式的目标和对象的技术。
  - 行人检测
  - 人脸识别
  - 手写识别
  - 车牌识别
  - 等等

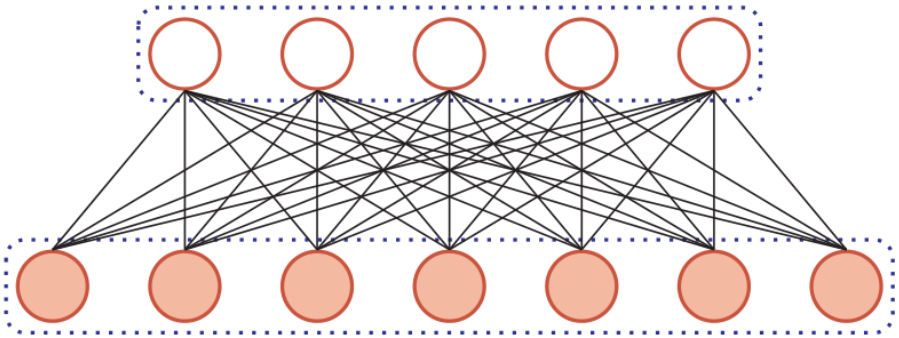
# MNIST写识别

- 参考

MINST手写识别.ipynb

# 卷积神经网络

- 经典BP神经网络权重矩阵的参数非常多



- 难以提取局部不变性特征
  - 自然图像中的物体都具有局部不变性特征，比如尺度缩放、平移、旋转等操作不影响其语义信息。
  - 而全连接前馈网络很难提取这些局部不变特征。

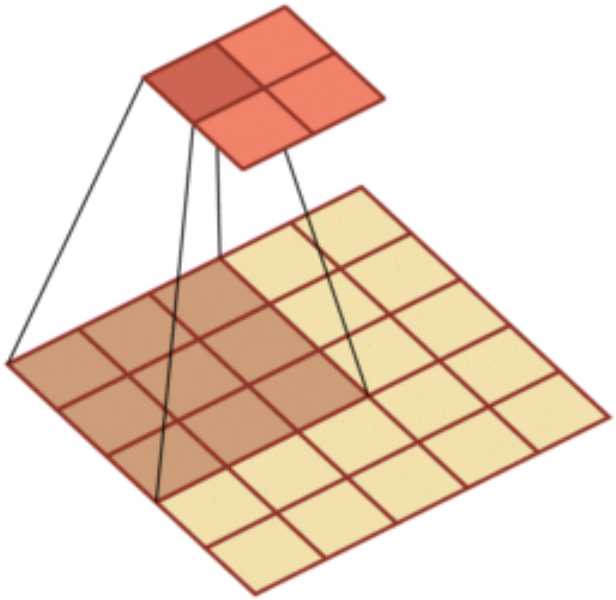
# 卷积神经网络

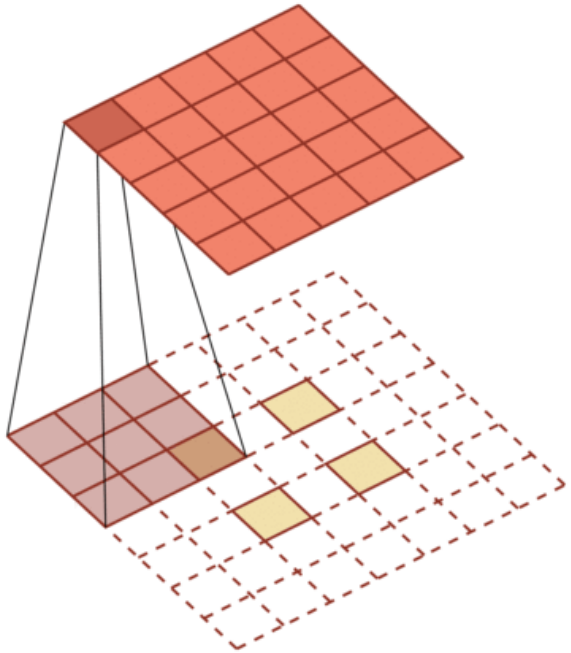
- 卷积神经网络（Convolutional Neural Networks，CNN）是一种前馈神经网络。
  - 卷积神经网络是受生物学上感受野（Receptive Field）的机制而提出的。
  - 在视觉神经系统中，一个神经元的感受野是指视网膜上的特定区域，只有这个区域内的刺激才能够激活该神经元。

- 卷积神经网络有三个结构上的特性：
  - 局部连接
  - 权重共享
  - 空间或时间上的次采样（下采样）

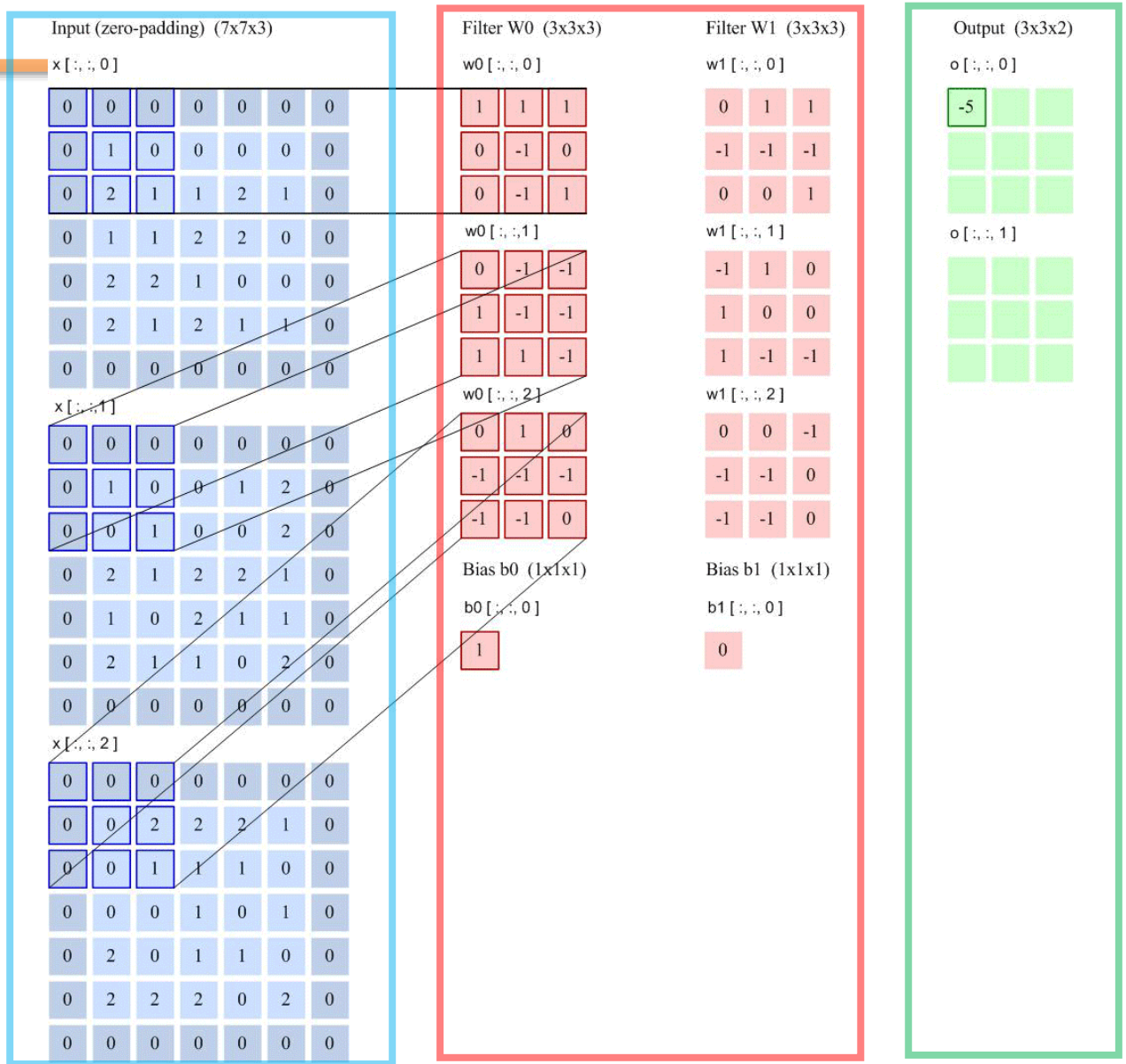
# 卷积神经网络

- 通过卷积实现低维特征映射到高维特征



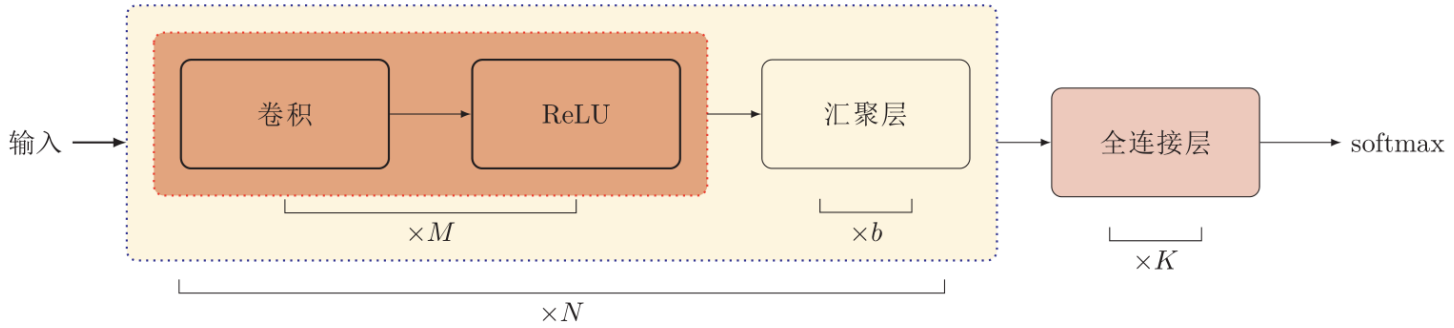


# 卷积神经网络


Input (zero-padding) (7x7x3)
x [:, :, 0]
0 0 0 0 0 0 0
0 1 0 0 0 0 0
0 2 1 1 2 1 0
0 1 1 2 2 0 0
0 2 2 1 0 0 0
0 2 1 2 1 1 0
0 0 0 0 0 0 0
x [:, :, 1]
0 0 0 0 0 0 0
0 1 0 0 1 2 0
0 0 1 0 0 2 0
0 2 1 2 2 1 0
0 1 0 2 1 1 0
0 2 1 1 0 2 0
0 0 0 0 0 0 0
x [:, :, 2]
0 0 0 0 0 0 0
0 0 2 2 2 1 0
0 0 1 1 1 0 0
0 0 0 1 0 1 0
0 2 0 1 1 0 0
0 2 2 2 0 2 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0 [:, :, 0]
1 1 1
0 -1 0
0 -1 1
w0 [:, :, 1]
0 -1 -1
1 -1 -1
1 1 -1
w0 [:, :, 2]
0 1 0
-1 -1 -1
-1 -1 0
Bias b0 (1x1x1)
b0 [:, :, 0]
1
Filter W1 (3x3x3)
w1 [:, :, 0]
0 1 1
-1 -1 -1
0 0 1
w1 [:, :, 1]
-1 1 0
1 0 0
1 -1 -1
w1 [:, :, 2]
0 0 -1
-1 -1 0
-1 -1 0
Bias b1 (1x1x1)
b1 [:, :, 0]
0
Output (3x3x2)
o [:, :, 0]
-5
o [:, :, 1]

# 典型结构

- 卷积网络是由卷积层、下采样层、全连接层交叉堆叠而成。
  - 趋向于小卷积、大深度
  - 趋向于全卷积
- 典型结构




  - 一个卷积块为连续M 个卷积层和b个汇聚层（池化层）（M通常设置为2～5，b为0或1）。一个卷积网络中可以堆叠N 个连续的卷积块，然后在接着K 个全连接层（N 的取值区间比较大，比如1～100或者更大；K一般为0～2）。

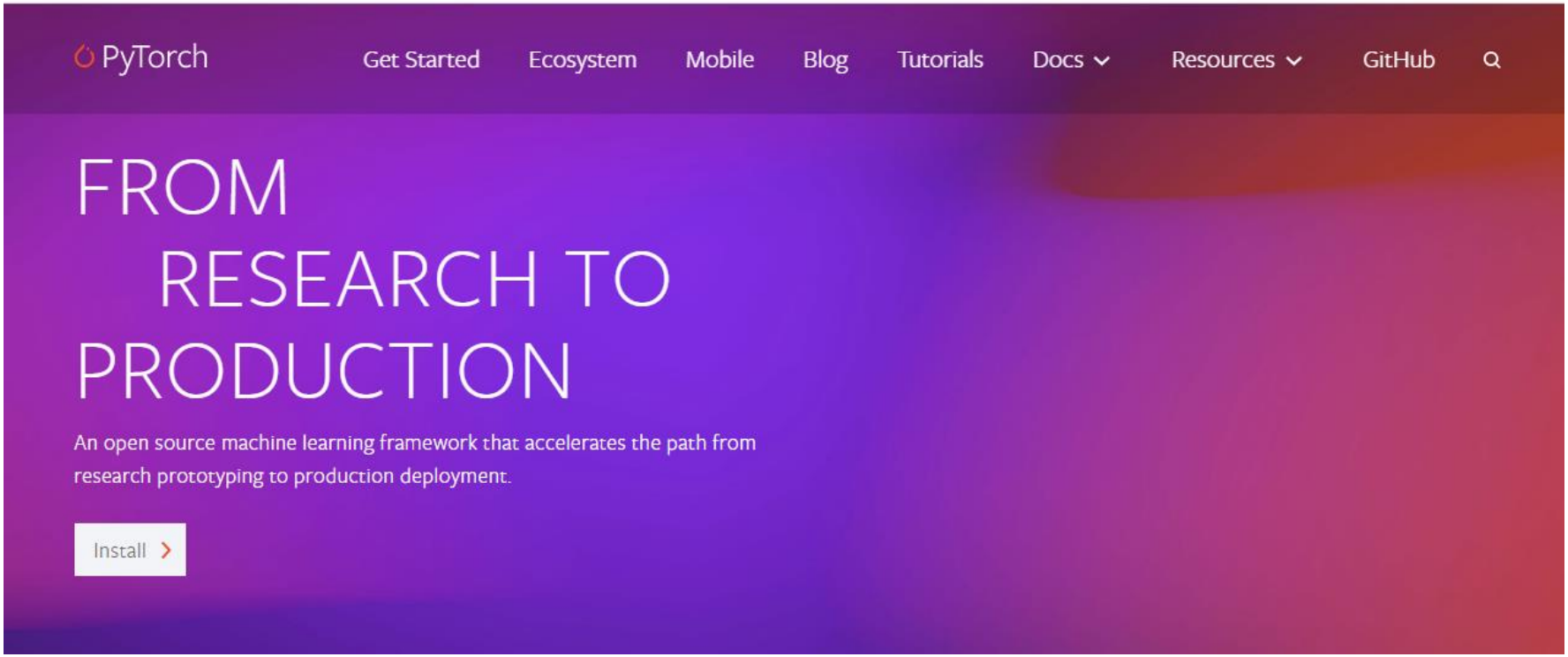
# 利用Pytorch进行图像识别

- PyTorch是由Facebook的人工智能研究团队开发的开源Python机器学习库，基于Torch，底层由C++实现，应用于人工智能领域。
- PyTorch的两大功能使其在深度学习领域大放异彩：
- 可使用GPU的张量计算库
- 自动微分系统

——可以简洁地定义计算图，损失函数，并选择优化器进行模型求解。

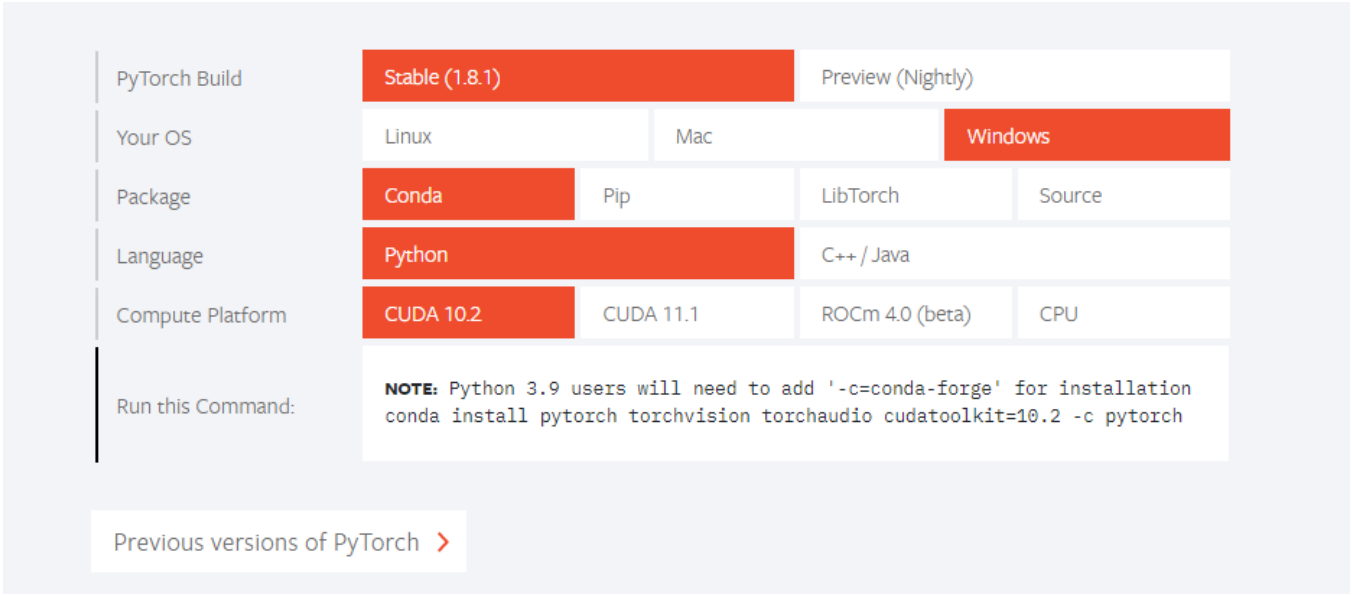
——内置了经典模型结构和模型参数，便于调用、微调。

# Pytorch安装


PyTorch
Get Started
Ecosystem
Mobile
Blog
Tutorials
Docs
Resources
GitHub
FROM
RESEARCH TO
PRODUCTION
An open source machine learning framework that accelerates the path from research prototyping to production deployment.
Install

# Pytorch安装

- 基本流程：
  - 安装Python（安装conda 和pip）
  - 确认Pytorch版本
  - 安装虚拟环境
  - 在虚拟环境中安装Pytorch




https://pytorch.org/

# 虚拟环境配置

- 配置虚拟环境，以便在jupyter notebook中访问

- 创建：conda create -n mytorch python=3.8
- 查看：conda env list
- 进入：conda activate mytorch
- 退出：conda deactivate

- 配置jupyter：
- conda install nb_conda
- conda activate 虚拟环境名字
- conda install ipykernel

# 基本使用

- 参考官方文档，

https://pytorch.org/tutorials/

- 以及

Pytorch-quickstart_tutorial.ipynb

# 练习

- 对MINST数据集的数据进行可视化，查看数据分布情况；
- 将MINST数据按照7:2:1的比例随机划分为训练集、验证集和测试集；
- 分别利用sklearn中的SVM、神经网络、以及Pytorch中的卷积神经网络（可选）对MINST数据进行分类识别，并比较模型的运行速度和效果。
- （模型具体参数设定不做限制）

谢谢！