**1. 图像表示**

* 图像在python中使用矩阵表示
  + Numpy模块的ndarray
* OpenCV：
  + C++ : cv::Mat
  + Python: numpy.ndarray
* OpenCV
  + imread：读取图像
  + imwrite:保存图像

1. 图像的矩阵表示

In [ ]:

**import** **numpy** **as** **np**

**import** **cv2**

*# 图像 既 矩阵*

img\_py = [[[0, 255, 0, 120] **for** i **in** range(255)] **for** j **in** range(255)]

img\_np = np.array(img\_py)

cv2.imwrite("1.png", img\_np)

* OpenCv的图像的格式：
  + BGR[A]
* PyTorch:
  + [N C H W]：NCHW格式
* Qt的图像（Window）
  + RGB[A]

1. 读取已知图像

In [ ]:

**import** **cv2**

img = cv2.imread("gpu.bmp")

print(img.shape)

1. 图像基本操作

* 转置

In [ ]:

**import** **cv2**

img = cv2.imread("gpu.bmp")

print(img.shape)

print(img.T.shape)

img\_t =img.transpose(2, 0, 1)

print(img\_t.shape)

* 改变形状
  + 只能改变形状，不能改变数据
    - 改变前的元素个数 = 改变后的元素格式

In [ ]:

**import** **cv2**

img = cv2.imread("gpu.bmp")

img\_shape = img.reshape(800, img.shape[0] \* img.shape[1]//800 ,3)

cv2.imwrite("2.bmp", img\_shape)

* 改变大小
  + resize
    - numpy.resize
      * 把矩阵拉抻为向量，在根据大小截断，然后用截断的数据形成图形。
    - cv::Mat::resize
      * 缩放图像（被覆盖）
      * numpy.resize()
    - cv::resize
      * 缩放图像

In [ ]:

**import** **cv2**

img = cv2.imread("gpu.bmp")

*# img.resize([600, 800, 3])*

img = cv2.resize(img, (600, 800))

cv2.imwrite("3.jpg", img)

* 缩维/扩维
  + 不影响数据
    - 缩维只能对长度为1的维度进行缩放。

In [ ]:

**import** **cv2**

**import** **numpy**

img = cv2.imread("gpu.bmp")

*# 扩维：numpy，opencv的方式*

img\_ex = numpy.expand\_dims(img, axis=1)

print(img\_ex.shape)

img\_ex2 = img[:,:,:, numpy.newaxis]

print(img\_ex2.shape)

In [ ]:

img\_sq = img\_ex.squeeze(axis=1)

print(img\_sq.shape)

1. 几何图形的绘制
   * putText:文本绘制
   * reactangle：矩阵绘制

In [ ]:

**import** **numpy**

**import** **cv2**

*# 构造一个缓冲*

img = numpy.ndarray(shape=(400,400, 3), dtype=numpy.uint8)

img.fill(180) *# 填充背景*

*# 绘制*

cv2.rectangle(img, [100, 100, 200, 200], [0, 0, 255], 2)

cv2.circle(img, (200, 200), 100, [0, 255, 0], 2)

cv2.putText(img, "OpenCV", (100, 200), cv2.FONT\_HERSHEY\_SIMPLEX, 2.0, [255, 0, 0], 2)

cv2.imwrite("4.jpg", img)

**2. 图像特征**

**2.1. 卷积运算**

* 卷积运算公式：
  + h(x)=∫+∞−∞f(τ)g(x−τ)dτh(x)=∫−∞+∞f(τ)g(x−τ)dτ
* 离散方式：
  + x→(x,y)x→(x,y)
  + τ→(i,j)τ→(i,j)
  + ∫→∑∫→∑
  + h(x,y)=∑(i,j)∈[−N,N]f(i,j)g(x−i,y−j)h(x,y)=∑(i,j)∈[−N,N]f(i,j)g(x−i,y−j)
* h(x,y)=∑(i,j)∈[−1,1]f(i,j)g(x+i,y+j)h(x,y)=∑(i,j)∈[−1,1]f(i,j)g(x+i,y+j)
* 反卷积运算
  + f(i,j),(i,j)∈[−N,N]f(i,j),(i,j)∈[−N,N] 称为卷积核
  + g(x,y),(x,y)∈[−∞,+∞]g(x,y),(x,y)∈[−∞,+∞] 图像
  + h(x,y)h(x,y)称为特征图(Feature Map)
* 提供了一个函数来做卷积运算：
  + 高斯模糊
    - Gaussblur
  + 卷积运算
    - filter2D

In [ ]:

**import** **cv2**

img = cv2.imread("gpu.bmp")

img\_gauss = cv2.GaussianBlur(img, (15, 15), 10)

cv2.imwrite("5.jpg", img\_gauss)

In [ ]:

**import** **cv2**

**import** **numpy**

img = cv2.imread("gpu.bmp")

*# kernel = numpy.ndarray(shape=(9, 9), dtype=numpy.float)*

*# kernel.fill(1.0/81)*

*# kernel= numpy.array(*

*# [*

*# [-1, 0, 1],*

*# [-1, 0, 0],*

*# [-1, 0, 1]*

*# ]*

*# )*

*# img\_filter2D = cv2.filter2D(img, -1, kernel, delta= 0)*

img\_filter2D = cv2.Sobel(img, -1, 2, 2, ksize = 3, scale = 1.0, delta=1.0)

cv2.imwrite("9.jpg", img\_filter2D)

**2.2. 图像特征的定义**

* 定义的方式：
  + 统计定义
  + 微积分定义
  + ....
* 图像特征：
  + 变化的部分才是图像特征
  + 像素的变化（像素的梯度）
    - 斑点
    - 边界
    - 角点
  + 数学原型：
    - 微分/求导
    - 一阶导数： Sobel梯度
    - 二阶导数： Laplace梯度
  + 一个图像进行求导运算，得到梯度特征图
    - 怎么对图像求导
* 图像的求导模型
  + f′(x)=limϵ→0f(x+ϵ)−f(x−ϵ)2ϵf′(x)=limϵ→0f(x+ϵ)−f(x−ϵ)2ϵ
  + ϵ=1ϵ=1
* f′(x)≈f(x+1)−f(x−1)2f′(x)≈f(x+1)−f(x−1)2
* f′+(x)≈f(x+1)−f(x)+0×f(x−1)f+′(x)≈f(x+1)−f(x)+0×f(x−1)
* 从方程表示方式到代数向量表示方式
  + 公式表示（略）
  + 图像的梯度计算（图像微分）可使用卷积运算表示。
* 偏导数（1rank / 1 rank）的融合：
  + 和
  + 1范数：
  + 2范数

**2.3. 图像特征的计算**

In [ ]:

**import** **cv2**

**import** **numpy**

img = cv2.imread("gpu.bmp")

*# kernel = numpy.ndarray(shape=(9, 9), dtype=numpy.float)*

*# kernel.fill(1.0/81)*

*# kernel= numpy.array(*

*# [*

*# [-1, 0, 1],*

*# [-1, 0, 0],*

*# [-1, 0, 1]*

*# ]*

*# )*

*# img\_filter2D = cv2.filter2D(img, -1, kernel, delta= 0)*

img\_filter2D = cv2.Sobel(img, -1, 2, 2, ksize = 3, scale = 1.0, delta=1.0)

cv2.imwrite("9.jpg", img\_filter2D)

**3. 图像特征学习**

* 卷积运算对图像的大小的影响
  + 经过卷积运算图像变小：
    - 原图像大小：SS
    - 卷积核大小：KK
    - 目标图像大小：D=S−K+1D=S−K+1
* 32,5→2832,5→28
* 假设想目标图像大小=原图像
  + 对原图像补边：padding = K−12K−12
* 图像降维：
  + 池化运算：pool
    - 均值
    - 最大池化：（推荐）
  + 负像素：
    - 过滤（负的像素值设置为0）
    - 激活函数activity function。
* MNIST类

In [ ]:

**import** **torchvision**

**import** **torch**

*# help(torch.utils.data.dataset.Dataset)*

help(torchvision.datasets.MNIST)

* 总结：
  + 卷积神经网络： 图像特征抽取 + 分类器
    - 训练的图像的卷积核参数
    - 训练的分类器的参数
    - 梯度下降法：
* 作业：
  1. 把上课的例子程序跑起来。
  2. 把测试集加载，常识验证测试集的识别的准确。