SLDS 2025: Project 1

Introduction to Project and Background Knowledge

2025.4.7

SUSTech Southern University of Science and Technology

Project 1：图像分类任务探究

SUSTech Southern University of Science and Technology

- 手写数字识别是图像识别领域的经典问题，旨在通过对包含手写数字的图像数据进行分析，让模型能够准确判断出图像中所写的数字是 0 - 9 中的哪一个。本项目的目标是构建一个有效的手写数字识别算法或模型，并深入探究数据相关因素对算法或模型性能的影响，同时掌握避免过拟合的方法。

- 在Project 1中，同学们需要运用机器学习和深度学习的方法对MNIST手写数字数据集中的手写数字进行识别，我们建议各位同学使用Python程序设计语言及其相关库（如NumPy, Pandas, scikit-learn等）来实现Project 1的各项任务。

- 项目提交内容:
  - 一份包含完整Python代码的Jupyter Notebook或.py文件，其中应包括注释以解释代码的作用。
  - 一份简要的PDF项目报告，总结为完成Project 1中各个Task所使用方法的目的、方法本身、结果和最终的结论。

Project 1：图像分类任务探究

SUSTech Southern University of Science and Technology

- Task 1：数据预处理
  - 使用Pandas、PIL或cv2等库实现对数据的读取，并进行初步的观察和处理，例如维度转换等操作。
  - 尝试对图像数据进行简单的统计学特征提取，例如平均像素值、二值化后像素值占比等，并探究能否从其中找到有利于分类任务的显著特征。

- Task 2：数据可视化
  - 使用t-SNE、UMAP等数据降维方法，结合Matplotlib等库对手写数字图像进行可视化，从而展示数据集的样本分布，并进一步阐释从中观察到的现象和造成这些现象的可能原因。
  - 使用如添加高斯噪音、随机翻转、随机裁剪等部分数据增强方法处理原始数据集，并做对应的样本分布可视化，并分析这些数据增强方法的有效性及其原因。
  - 注意不是所有的数据增强方法都适用于MNIST，但是我们鼓励对效果不好的方法进行原因探究。

Project 1：图像分类任务探究

SUSTech Southern University of Science and Technology


mnist_train_0.jpg


mnist_train_1.jpg


mnist_train_2.jpg


mnist_train_3.jpg


mnist_train_4.jpg


mnist_train_5.jpg


mnist_train_6.jpg

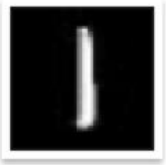

mnist_train_7.jpg


mnist_train_8.jpg


mnist_train_9.jpg


mnist_train_10.jpg


mnist_train_11.jpg


mnist_train_12.jpg


mnist_train_13.jpg


mnist_train_14.jpg


mnist_train_15.jpg


mnist_train_16.jpg


mnist_train_17.jpg


mnist_train_18.jpg


mnist_train_19.jpg

Project 1：图像分类任务探究

SUSTech Southern University of Science and Technology

- Task 3：分类模型训练实践
  - 按照MNIST数据集官方提供的训练集\测试集划分，分别使用多种机器学习和深度学习方法训练分类模型并进行测试，获得相应的精确率指标并进行比较。
  - Task 3.1：常见的用于描述模型性能的指标有哪些？分别是如何计算的？有什么实际意义？
  - Task 3.2：机器学习方法需要至少选择3种（Decision Tree、Random Forest、SVM、逻辑回归等），比较各种方法的各项性能指标，分析各种方法在MNIST数据集上表现差异的可能原因。
  - Task 3.3：深度学习方法仅要求实现CNN，比较CNN模型和机器学习模型的性能指标差异，并通过实验探究超参数和网络深度、网络结构等对CNN性能的影响。

Project 1：图像分类任务探究

SUSTech Southern University of Science and Technology

- Task 4：数据数量的平衡探讨
  - 初始探索：先使用MNIST中的部分数据进行实验，比如先选取 1000 张训练图像（每个数字类别均匀选取 100 张左右）和200张测试图像，观察各算法模型的初始准确率和收敛情况，了解模型在少量数据下的表现。
  - 逐步增加数据量：逐步增加训练数据的数量，每次增加1000至2000张，重新训练和评估模型，记录准确率等性能指标的变化情况。可以绘制数据量与准确率的关系曲线，观察随着数据量增多，准确率提升的趋势是否逐渐变缓，分析合适的数据量规模。
  - 对比不同量级数据：使用不同量级的数据训练不同版本的模型，比如一个用 1000 张训练图像训练的模型、一个用 10,000 张训练图像训练的模型和一个用全部MNIST训练图像（60,000 张）训练的模型，对比它们在相同测试集上的准确率、召回率等指标，直观感受数据量对最终结果的影响。

Project 1：图像分类任务探究

SUSTech Southern University of Science and Technology

- Task 5：欠拟合与过拟合现象探究
  - 在进行分类任务时，尝试记录各算法分类错误的样本，并在样本降维的可视化中加以体现。分析不同算法分类错误的样本集合是否具有相似性？如果有，这些样本有着怎样的特点？试探究这些样本难以被分类正确的原因。
  - 在深度学习网络模型的训练中，过拟合现象时有发生，这会比较严重地影响训练效率和最终预测的效果，试探究一些避免过拟合的方法对模型训练效果的影响：

Project 1：图像分类任务探究

SUSTech Southern University of Science and Technology

- Task 5：欠拟合与过拟合现象探究
  - L1和L2正则化：在模型的损失函数中添加L1或L2正则化项，通过调整正则化参数的大小，控制模型的复杂度，防止权重过大，使得模型能够学习到更具泛化能力的特征表示。观察不同正则化强度下模型在验证集和测试集上的性能变化。
  - Dropout：在神经网络的训练过程中，随机地将一部分神经元的输出置为0，减少过拟合现象。在不同的网络层应用不同的 Dropout 概率进行实验。
  - 选择合适复杂度的模型：对比不同复杂度的模型（如简单的感知机模型、浅层神经网络、深层神经网络等）在相同数据集上的表现，选择一个既能够拟合数据又不会过于复杂导致过拟合的模型结构。例如，发现深层神经网络在训练集上准确率很高但在测试集上下降明显时，考虑简化网络层数或者采用一些轻量级的网络架构。

Project 1：图像分类任务探究

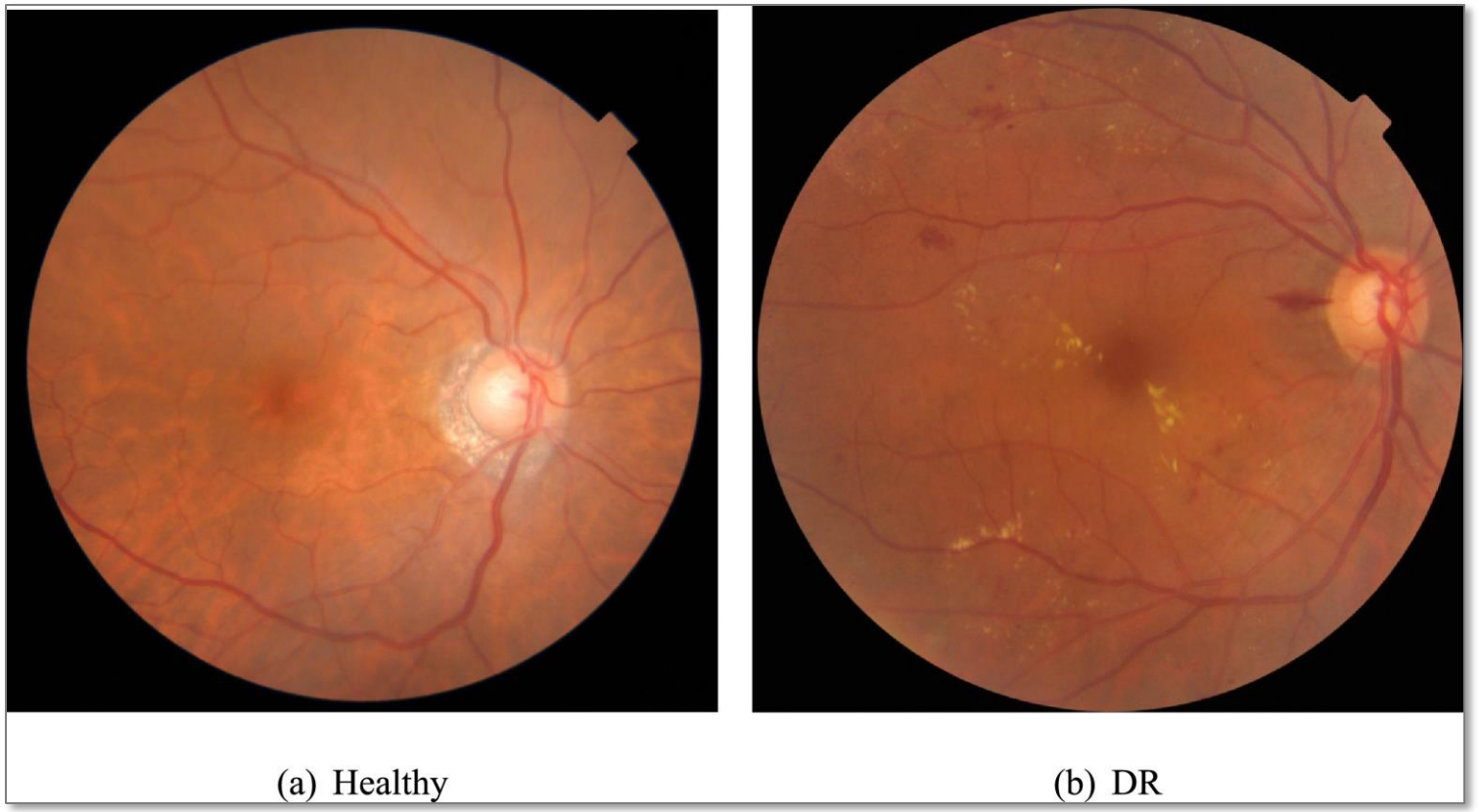
SUSTech Southern University of Science and Technology

- Bonus Task 1：图像聚类任务探究
  - 使用至少两种聚类方法对MNIST数据集中的图像样本进行聚类，将聚类结果在图像分布可视化结果中进行呈现，观察聚类结果和各样本真实标签是否存在一致性。
  - 将同样的聚类方法应用于CNN中的各个卷积层输出（有多个通道的可以进行合并），观察这些卷积层输出的聚类效果与各样本真实标签的一致性变化趋势，如果发现了一些规律，可以总结并尝试分析原因。

- Bonus Task 2：糖尿病视网膜病变图像分级任务
  - 将MNIST图像数据集更换为DDR眼底彩照图像数据集，然后使用相同的算法或模型进行训练和评估，观察性能指标是否存在明显的差异。
  - DDR眼底彩照图像数据集对应的具体任务是糖尿病视网膜病变的分级（共5等级），可视作分类任务。
  - 对于MNIST上尝试过的各种图像增强方法，在DDR上也进行尝试，观察模型的训练和预测效果，如果有些方法在两种数据集上存在明显效果差异，尝试分析其可能的原因。

Project 1：图像分类任务探究

SUSTech Southern University of Science and Technology



(a) Healthy　　(b) DR

从机器学习出发

SUSTech Southern University of Science and Technology

- 试图找到某种变换过程，将自变量和因变量对应起来，或者说是找到某种**函数**

- 但是很多传统机器学习方法在处理实际复杂问题的时候效果不太好
  - 依赖特征工程和算法选择，适用于结构化数据
  - 参数量相对少且可以分析具体含义（可解释性好）
  - 实际问题通常是难以建模、难以人工提取特征、数据维度高、数据体量大的

从机器学习出发

SUSTech Southern University of Science and Technology

- 那么有没有不怎么依赖人工特征选择，能自动学习特征，且处理复杂问题效果不错的方法呢?
- 有的兄弟，有的，那就不得不说基于深度神经网络的深度学习方法了。
- 深度神经网络的特点就是参数量大，因此训练需求的数据规模大、算力成本也高
  - 试想一个简单的多项式回归拟合模型，对于n次模型（n+1个多项式系数要求）我们至少要有n+1个数据点才能预测的比较准确
  - 深度神经网络动不动以Million计的参数，LLM更是以Billion计，那要想拟合的好需要的数据量就少不了
- 相对的，由于参数太多了，所以深度神经网络的可解释性也相对差，很多时候你搞不清楚他到底学到了啥

神经网络

SUSTech Southern University of Science and Technology

- 神经网络可以看成是一种函数f，使得给定自变量x，能够给出最接近因变量y的预测值y’
- f=g(wx+b)，其中g是激活函数，w是权重，b是bias
  - 激活函数使得神经网络具备拟合非线性实际函数的能力，增强泛化性
- 多自变量的形式：f=g(w1x1+w2x2+w3x3+b)
- 多因变量的形式：f1=g(w1x+b1)，f2=g(w2x+b2)，f3=g(w3x+b3)

神经网络

SUSTech Southern University of Science and Technology

- 但是这个式子看起来一点也不“深度”，并且拟合不了什么复杂函数

- 那么介绍深度神经网络的绝活：加层变深：
  - f=g(w1x1+w2x2+b)
  - f=g(w3g(w1x1+w2x2+b1)+b2)
  - f=g(w4g(w3g(w1x1+w2x2+b1)+b2)+b3)

- 多层神经网络本质就是把上一层的输入作为下一层的输入用另一套权重w和b再算一遍

神经网络

SUSTech Southern University of Science and Technology

- 神经网络的层数一上去了，参数量就变大了，（一般情况下）拟合效果也就变好了
- 神经网络需要大量的数据来训练，实质是用大量x和y的值推算网络中所有参数w和b的值
- 可视化理解神经网络的网站：https://alexlenail.me/NN-SVG/

训练神经网络

- 训练神经网络需要挑选一个损失函数（Loss Function），使得训练过程中其值不断降低

$$w = w - \eta \frac{\partial L(w,b)}{\partial w}$$

$$b = b - \eta \frac{\partial L(w,b)}{\partial b}$$

学习率　　梯度

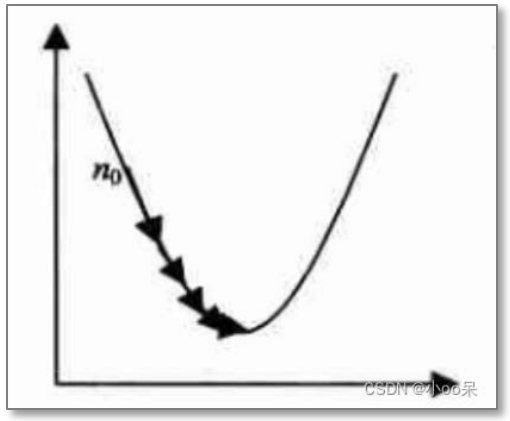
- 梯度下降：沿梯度下降最快的方向调整参数的值，使得下一次计算损失函数的值期望变小

训练神经网络

- 学习率（Learning Rate）一般调得小一点，而且越往后调得越小，这是为了避免“下降过头”

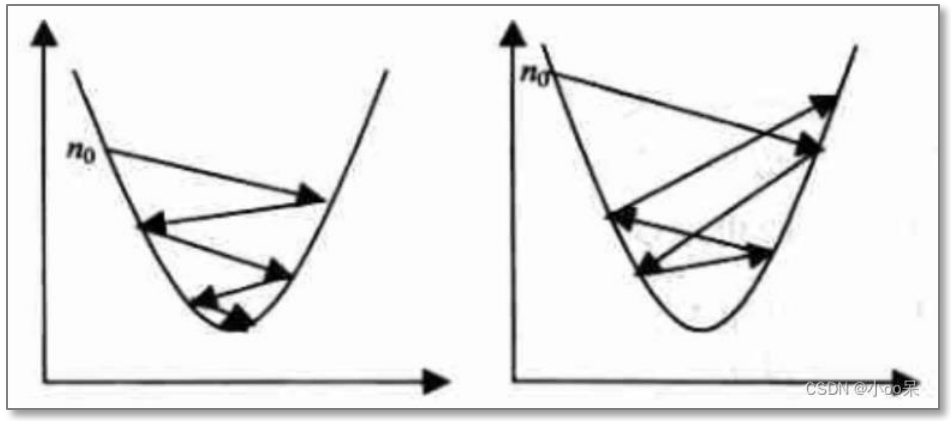
- 当然也不能调的太小了，那样训练速度就太慢了

学习率太小



找到最优解太慢

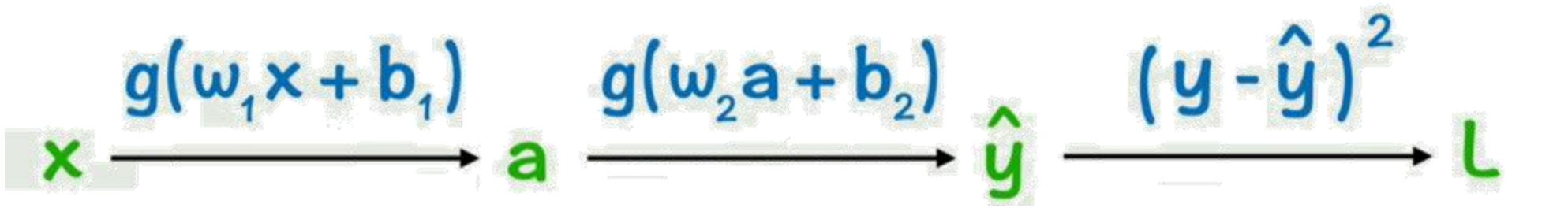
学习率过大



找到最优解太慢　　无法收敛

训练神经网络

- 深度神经网络有多层嵌套的参数，梯度下降的计算遵循反向传播的链式法则



链式法则求导 $\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial w_1}$ 从输出层反向传播

- 可视化理解反向传播和神经网络训练过程的网站：https://playground.tensorflow.org/

避免过拟合

SUSTech Southern University of Science and Technology

- 深度神经网络天然具有较大的参数量，所以如果需要拟合的目标函数其实复杂度不高就有可能出现过拟合
  - 过多的参数把噪声等干扰因素也学习到了

- 避免过拟合的方法：
  - 增加训练用数据量，使得训练数据在高维空间中的分布更均匀
  - 添加正则化惩罚项来遏制梯度下降时参数的过度变化趋势
  - 随机丢弃神经网络中的部分参数，即Dropout

避免过拟合

- 在损失函数中添加惩罚项的方法成为正则化，其目的是使得参数变化时考虑潜在的过拟合因素

$$L = L + \lambda \sum_{i=1}^{N} |w_i|$$ L1正则化项

$$L = L + \lambda \sum_{i=1}^{N} w_i^2$$ L2正则化项

正则化系数

- Dropout就简单了，是为了避免模型过于依赖部分参数从而过拟合的情况，可以更均衡地训练更多的参数

图像处理基础

- 图像可以看作是以单个像素为基本数据单元，由矩阵形式存储的高维数据形式

$$\boldsymbol{F}=\begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1N} \\ f_{21} & f_{22} & \cdots & f_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ f_{M1} & f_{M2} & \cdots & f_{MN} \end{bmatrix}$$

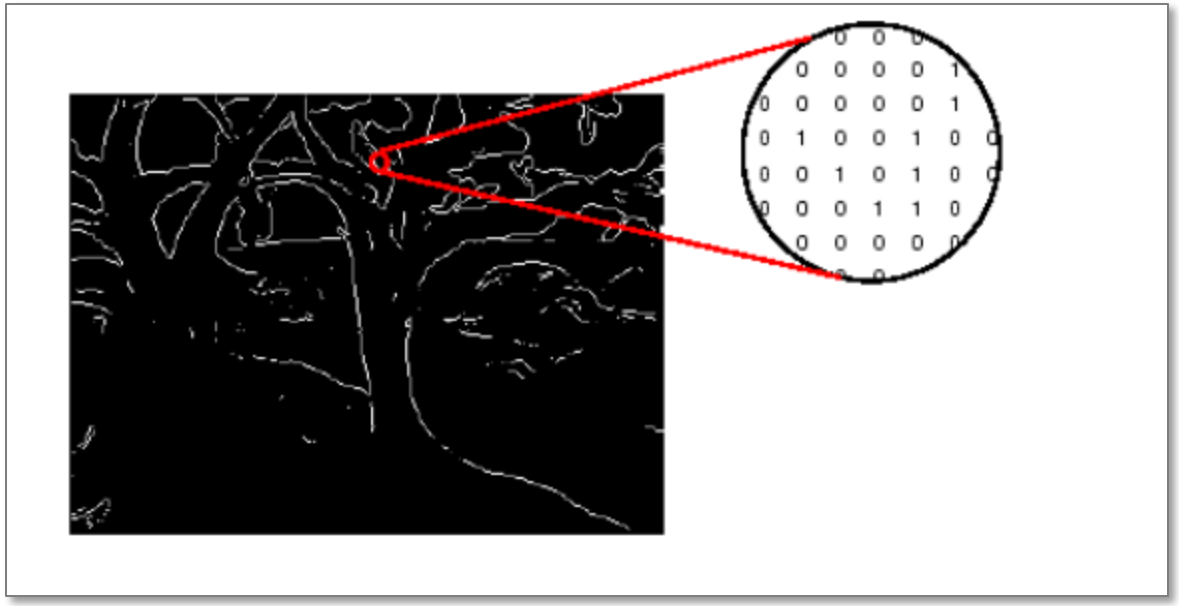
单个元素 f(x, y) 称为像素

- 图像可以按照多种标准进行分类：
  - 静止图像（JPEG、PNG）和运动图像（GIF）
  - 灰度图像（单通道）和彩色图像（RGB三通道）
  - 二维图像，三维图像和多维图像

图像处理基础

SUSTech Southern University of Science and Technology

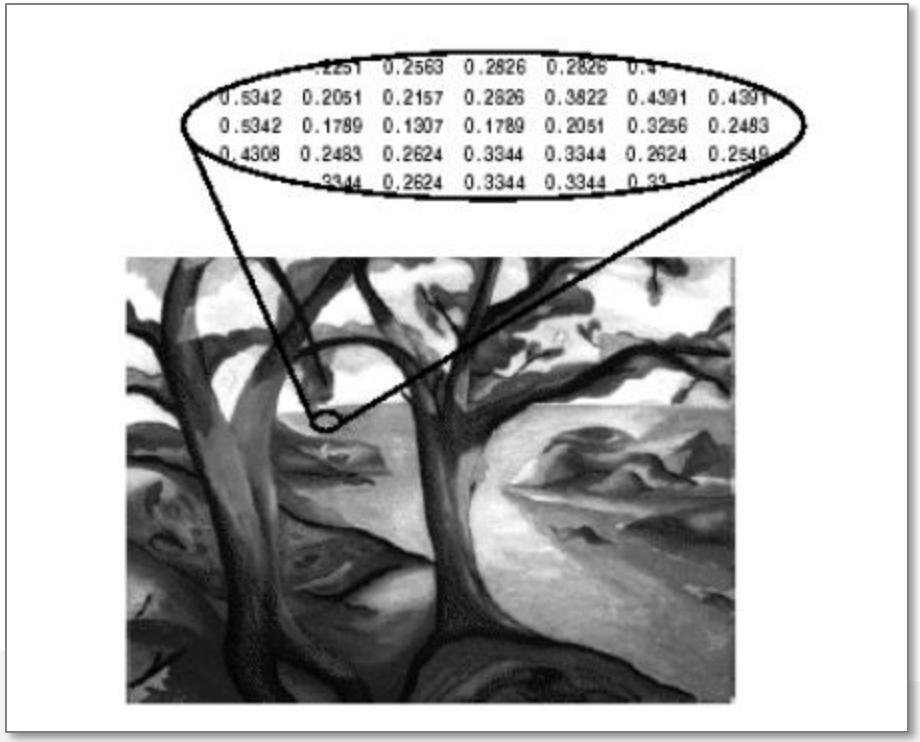
- 二值图像只有黑白两种颜色，0表示黑，1表示白，也有相反的表示方法，具体要看图像数据的说明。

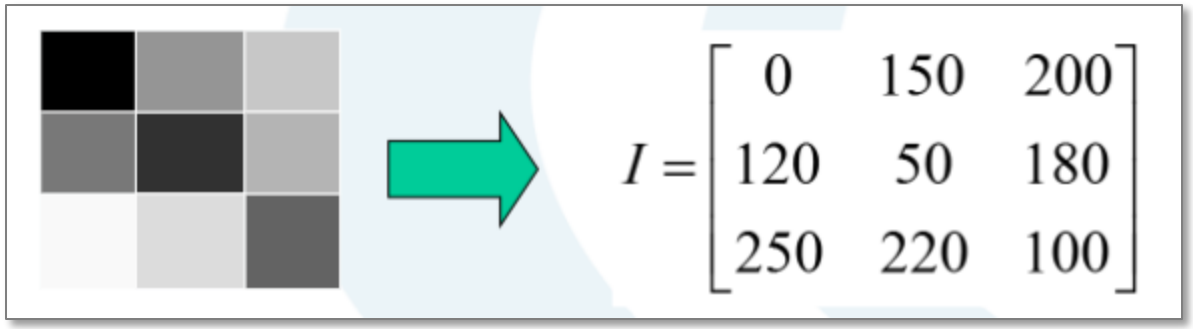


图像处理基础

- 灰度图像，也称为亮度图像或强度图像，通常用8bits整数表示（0-255）

- 每个像素都是介于黑色和白色之间的256种灰度中的一种，当然也有些图像特立独行不用0-255这个范围的





图像处理基础

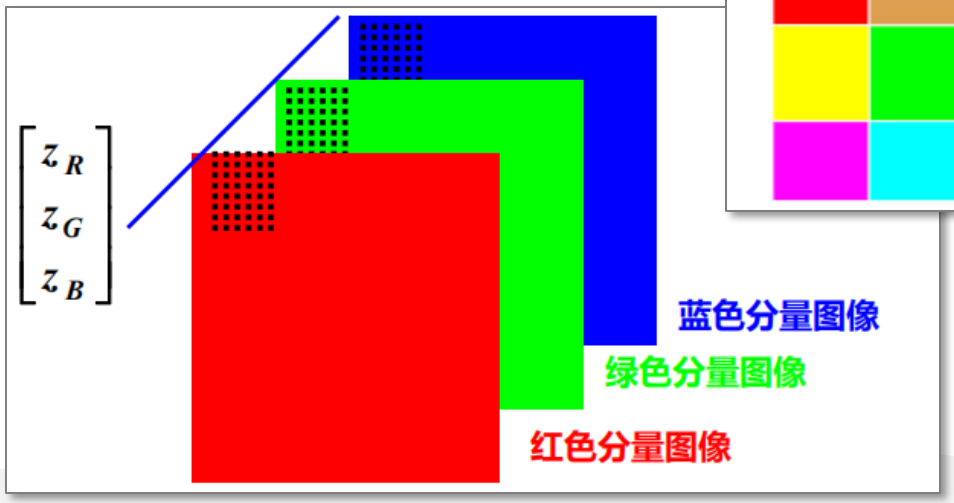
- 索引图像要特殊一点，其颜色是预先定义的（索引颜色）
  - 在索引值范围遵循0-255范围的情况下索引颜色的图像最多只能显示256种颜色（真彩色图像是256^3种）
  - 由数据矩阵X和色彩映射矩阵M组成
  - 矩阵M是一个大小为L×3，取值在[0, 1]的浮点数的数组，其长度L同它所定义的颜色数目相等。

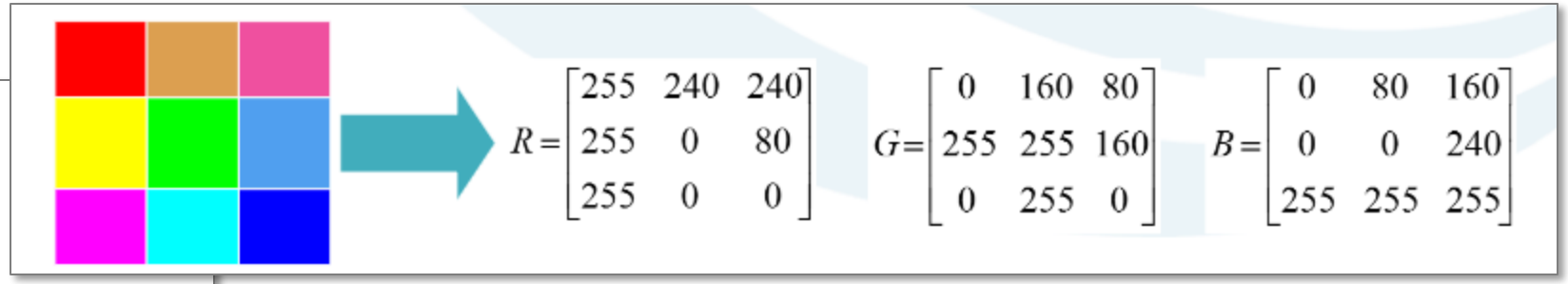
$$X=\begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix}_{m\times n} \qquad map=\begin{bmatrix} r_1 & g_1 & b_1 \\ r_2 & g_2 & b_2 \\ \vdots & \vdots & \vdots \\ r_i & g_i & b_i \\ \vdots & \vdots & \vdots \\ r_L & g_L & b_L \end{bmatrix}_{L\times 3}$$

图像处理基础

- RGB彩色图像具有三个通道，表示为一个M×N×3的数组，其中每一个彩色像素点都是在特定空间位置的彩色图像对应的红、绿、蓝三个分量

- 三个分量颜色通道中存储的值都为8bits整数，表示0到255之间的不同亮度值，总共可产生1670万种不同颜色





$$R=\begin{bmatrix}255 & 240 & 240\\ 255 & 0 & 80\\ 255 & 0 & 0\end{bmatrix}\quad G=\begin{bmatrix}0 & 160 & 80\\ 255 & 255 & 160\\ 0 & 255 & 0\end{bmatrix}\quad B=\begin{bmatrix}0 & 80 & 160\\ 0 & 0 & 240\\ 255 & 255 & 255\end{bmatrix}$$

图像处理基础

SUSTech Southern University of Science and Technology

图像	数字化后描述形式	备注
二值图像	f（X，Y）=1或0	文字、线条图、指纹等
黑白图像	$0 \le f(X,Y) \le 2^n-1$	黑白图像，一般n=6~8
彩色图像	$\lvert f_i(X,Y) \rvert$ i=R，G，B	以三基色表示的彩色图像
光谱图像	$\lvert f_i(X,Y) \rvert$ i=1，2…m	遥感图像，m=6~8或更大
立体图像	$f_l(X,Y)$，$f_r(X,Y)$	左L右R视点得到同物体的图像对
动态图像	$\lvert f_t(X,Y) \rvert$ t=t1，t2…tr.	动态图像，动画制做等

形态学图像操作

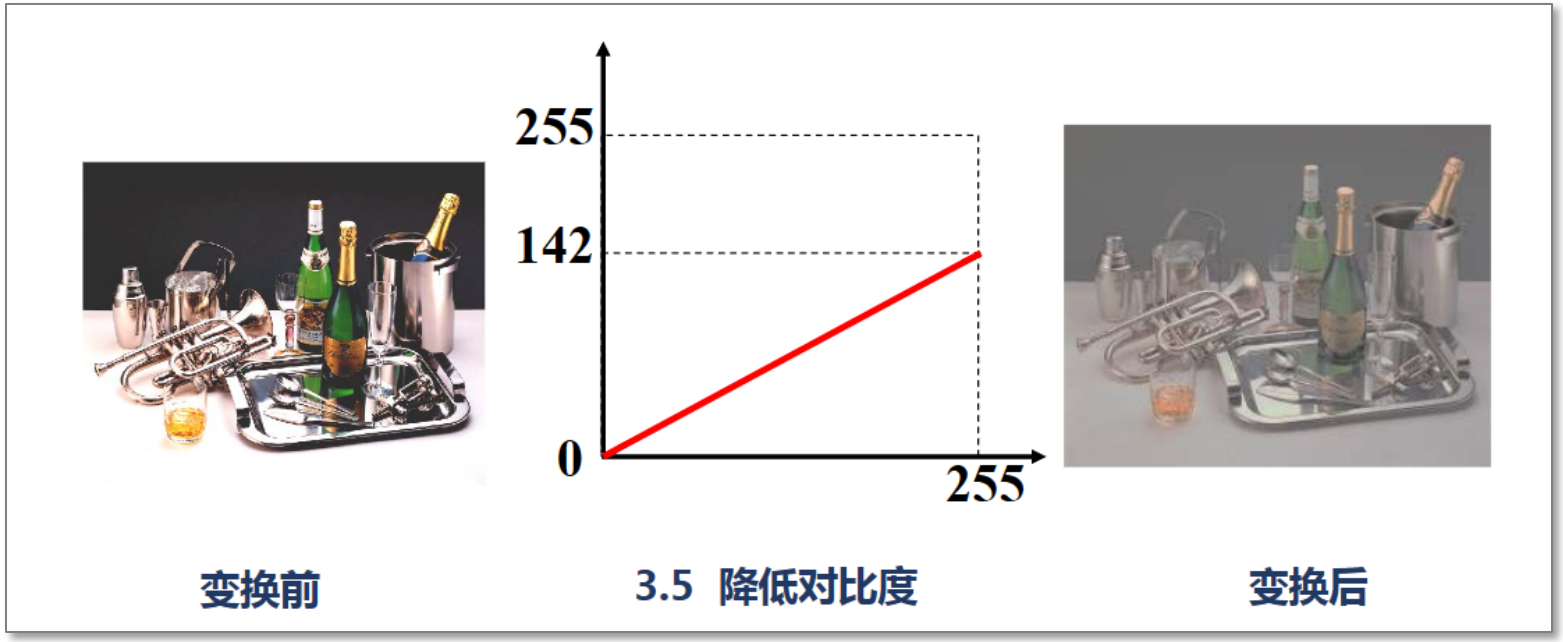
SUSTech Southern University of Science and Technology

- 形态学图像操作是最基本的图像修改与处理手段，常见的OpenCV库中就包含了大量不同的形态学操作

- 图像基本运算是进行形态学操作的基础，可以大致分为如下几类：
  - 点运算：二值化、取反色等
  - 代数运算：将两幅或多幅图像通过对应像素之间的加、减、乘、除运算
  - 逻辑运算：同上，但运算是逻辑与、或、非等
  - 几何运算：翻转、平移、放缩等

形态学图像操作

- 线性点运算一例：降低对比度

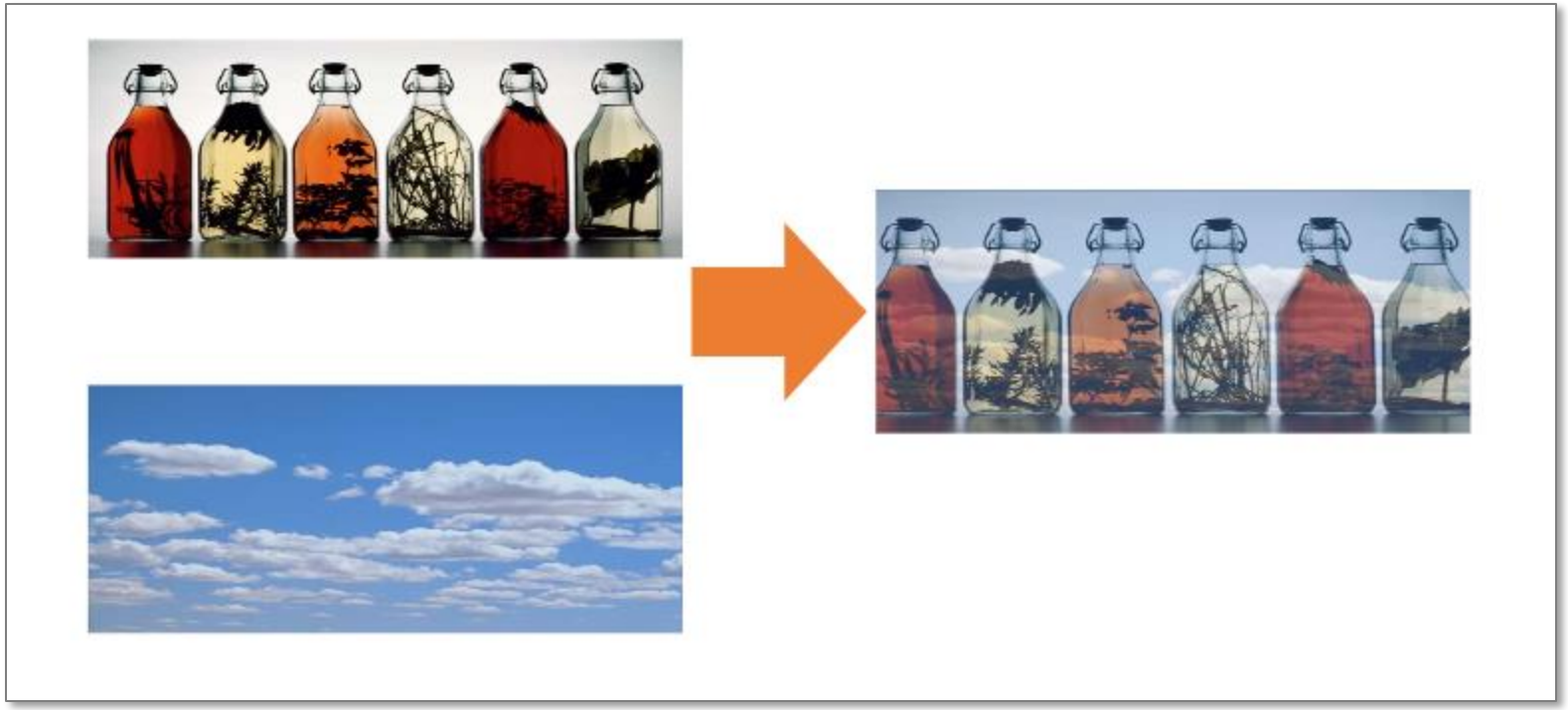


- 事实上像Photoshop这些图像编辑软件中的调色等功能都是点运算

形态学图像操作

SUSTech Southern University of Science and Technology

- 代数运算一例：图像叠加



形态学图像操作

SUSTech Southern University of Science and Technology

- 几何运算一例：图像镜像翻转

- 几何运算可以改变图像中物体对象（像素）之间的空间关系，从变换性质出发可分为：
  - 位置变换：平移、镜像、旋转等
  - 形状变换：放大、缩小等
  - 复合变换：上述变换的组合

形态学图像操作

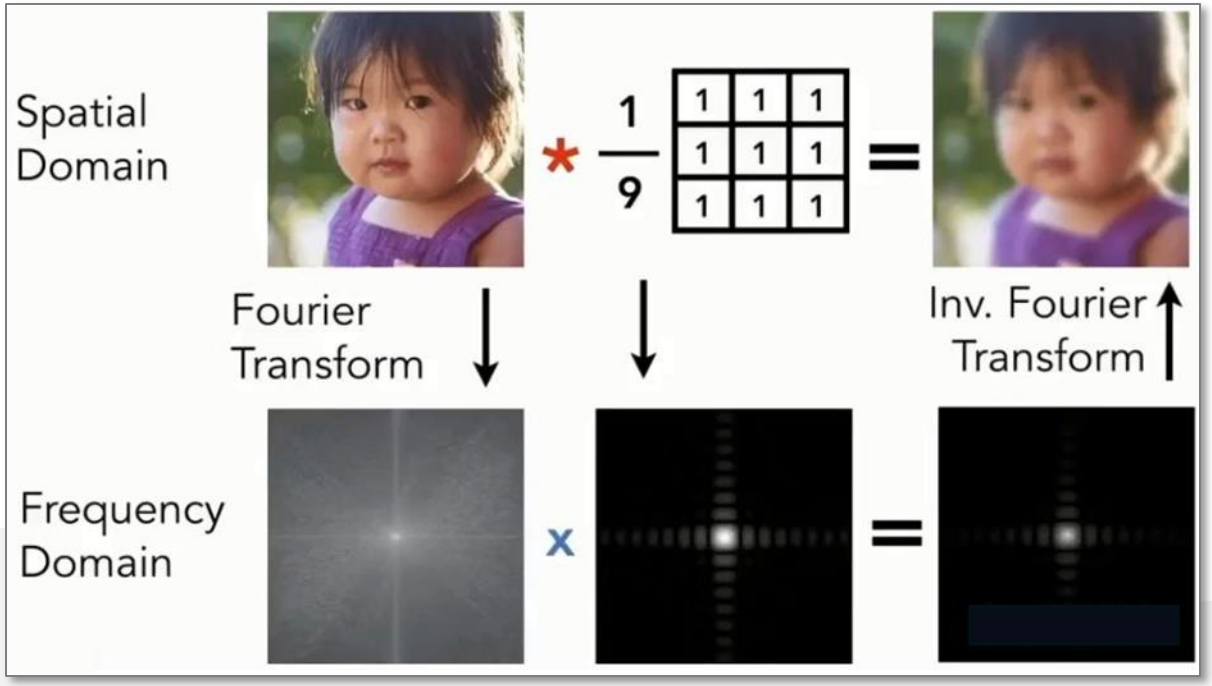
- 部分复杂的形态学操作不仅依赖像素自身的值，也与其周围像素的值有关
  - 黑白图像的膨胀操作：指定一个卷积核大小D，对每个像素进行计算，只要以该像素为中心D*D的范围内有有白色，那么该像素点膨胀操作之后就是白色
  - 黑白图像的腐蚀操作：指定一个卷积核大小D，对每个像素进行计算，只要以该像素为中心D*D的范围内有有黑色，那么该像素点腐蚀操作之后就是黑色

- 这些操作的组合可以实现很多有意义的功能：
  - 先腐蚀再膨胀：去除物体边缘毛刺
  - 先膨胀再腐蚀：填补物体内部缺漏
  - 用膨胀图像减去腐蚀图像：获取物体边缘轮廓

形态学图像操作

- 图像特征提取是对图像数据进行分类等操作的关键步骤

- 图像特征是图像中像素所具有的基本属性，从不同的角度描述了图像的性质
  - 基本的图像特征类别包括灰阶、颜色、形状、纹理、频域信息等



形态学图像操作

SUSTech Southern University of Science and Technology

- 刚才提到的都是一些图像的基本特征，之前也提到基本特征难以直接用于复杂任务

- 图像的高级特征是通过复杂的算法或深度学习模型提取的，能够捕捉图像的高层语义信息，这些特征通常用于更复杂的图像识别和分类等任务
  - 比如卷积神经网络（CNN）就可以通过预训练的深度学习模型（如VGG、ResNet等）提取图像隐含的高级特征，进而用于图像分类、检测和分割等任务
  - 有时候人类比较难理解和找出CNN具体得到了哪些高级特征，只能是CNN好使就先用着，所以我们说深度神经网络的可解释性没那么好

卷积神经网络（CNN）

SUSTech Southern University of Science and Technology

- 图像不是可表示为列向量的多个单一数值自变量，那么神经网络要如何处理图像这种二维数据呢?

- 一种朴素的想法是把二维矩阵展开成向量，然后直接往深度神经网络里面丢，但是这会有一些问题:
  - 破坏了图像中像素间的空间位置关联性
  - 展开生成的向量过长，导致每一层网络需要训练的权重参数过多

- 基于此，卷积被引入到神经网络中

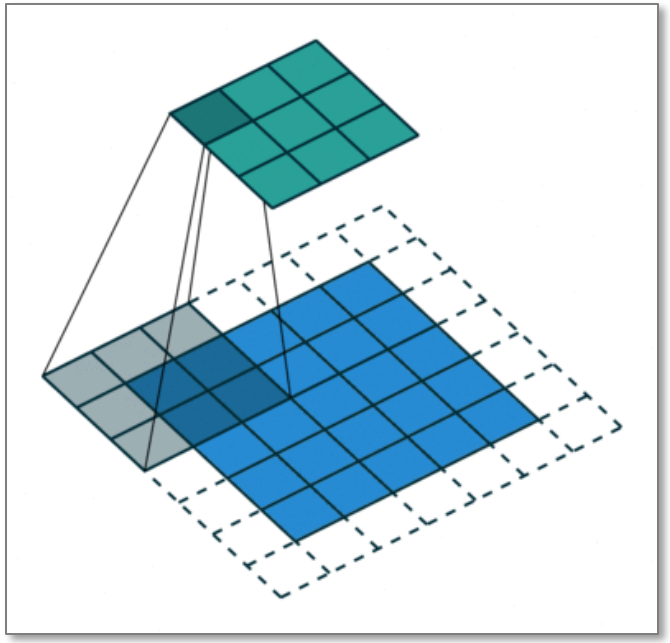
卷积神经网络（CNN）

SUSTech Southern University of Science and Technology

- 卷积是将使用一个固定大小卷积核对图像做平移卷积计算输出新矩阵的过程

- 刚才我们在一些形态学操作中也提到卷积核的概念，不过那些操作使用的都是参数已知的卷积核，深度神经网络中的卷积核参数是需要训练得到的

- 可视化理解CNN：https://poloclub.github.io/cnn-explainer/



卷积神经网络（CNN）

SUSTech Southern University of Science and Technology

- 一个卷积神经网络（CNN）结构中通常是如下层级的组合叠加：
  - 卷积层（Conv）：进行卷积操作的神经网络层
  - 池化层（Pool）：对上层卷积输出的中间特征矩阵进行降维并保留主要特征
  - 全连接层（FC）：此前介绍过的朴素神经网络层

- 可视化手写数字识别CNN网络推理过程：https://adamharley.com/nn_vis/cnn/3d.html

图像处理实践（Python）

SUSTech Southern University of Science and Technology

- cv2是Python编程语言的一个第三方库，提供对OpenCV的封装，从而可以方便的进行很多图像相关的操作
- 如遇到问题建议查阅cv2文档或者咨询大模型
- 图像增强之几何变换法：

```
# 图像旋转
import cv2
image = cv2.imread('image.jpg')
(h, w) = image.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, 45, 1.0)  # 旋转45度
rotated_image = cv2.warpAffine(image, M, (w, h))
```

图像处理实践（Python）

SUSTech Southern University of Science and Technology

- PIL是另一款Python编程语言中用于图像处理的库，也有很多强力的图像处理功能

- 图像增强之颜色变换法：

```
from PIL import Image, ImageEnhance

image = Image.open('image.jpg')

enhancer = ImageEnhance.Brightness(image)
image_bright = enhancer.enhance(1.5)  # 增加亮度

enhancer = ImageEnhance.Contrast(image)
image_contrast = enhancer.enhance(2.0)  # 增加对比度
```

图像处理实践（Python）

SUSTech Southern University of Science and Technology

- 增加随机噪声可以提高模型的健壮性，使其在有噪声干扰的数据上表现更好
- 噪声这块有很多选择：高斯噪声、椒盐噪声、泊松噪声、均匀噪声、斑点噪声等
- 图像增强之增加噪声法：

```
import numpy as np
#高斯噪声，均值为0、标准差为25
noise = np.random.normal(0, 25, image.shape).astype(np.uint8)
noisy_image = cv2.add(image, noise)
cv2.imshow('Noisy Image', noisy_image)
```

图像处理实践（Python）

SUSTech Southern University of Science and Technology

- 还可以随机擦除图像的一部分，模拟遮挡情况，以此提高模型在部分缺失数据上的表现

- 但是还是那句话，要看具体的图像语义情况，擦除的位置和大小要适当

深度学习代码实践（PyTorch）

SUSTech Southern University of Science and Technology

- PyTorch是专为Python编程语言提供的Torch库封装以及各种深度神经网络训练相关工具的合集，旨在加速深度学习模型的训练和推理过程，并可以利用GPU的CUDA API进行矩阵计算并行化加速

- 对于Project中使用CNN或者更复杂深度神经网络模型的场合，我们推荐使用PyTorch做工程实现

深度学习代码组成

1. Dataloader
2. Model
3. Loss
4. Hyperparameters
5. Training

1. Dataloader

```python
train_transform = transforms.Compose([
    transforms.Resize((28, 28)),
    # transforms.RandomCrop(),
    transforms.ToTensor(),
    transforms.Normalize([0.5],
                         [0.5],) # 由于创建的是1通道的，所以标准差和平均值都是一维
])

# 自定义数据集类
class myData(Dataset):
    def __init__(self, path, transform, kind):
        super(myData, self).__init__()
        self.path = path
        self.transform = transform
        self.data_info, self.labels = load_mnist(path, kind)

    def __getitem__(self, index):
        img = self.data_info[index]
        img = Image.fromarray(img) # transforms功能只能作用于img格式
        label = torch.tensor(self.labels[index], dtype=torch.long)
        if self.transform is not None:
            img = self.transform(img)
        return img, label

    def __len__(self):
        return len(self.data_info)
# 实例化数据集
trainset = myData(data_path,train_transform ,'train')
testset = myData(data_path,train_transform,'t10k')

train_loader = DataLoader(
    dataset=trainset,
    batch_size=100,
    shuffle=True
)
```

数据增强：

- 统一图像尺寸
- 添加噪音
- 随即裁剪

def_getitem_(self，index)：

1. 输入index，返回图像和对应标签

def_len_(self)：

2. 数据集长度

train_loader:

定义批处理大小batch_size

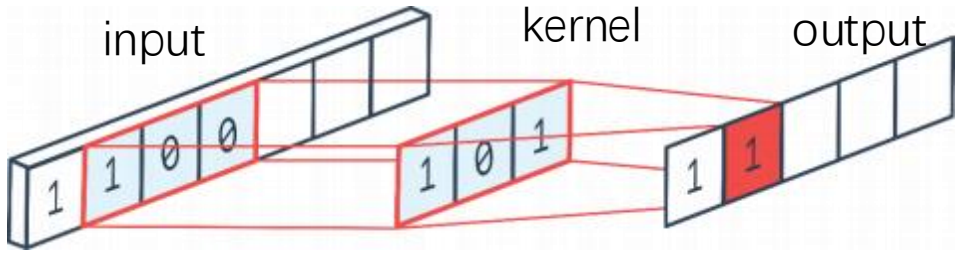
2. Model

2. self.conv11: conv1d

[64, 1, 2400] -> [64, 64, 2396]

```
class Doubleconv_33(nn.Module):
    def __init__(self, ch_in, ch_out):
        super(Doubleconv_33, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv1d(ch_in, ch_out, kernel_size=3),
            nn.ReLU(inplace=True),
            nn.Conv1d(ch_out, ch_out, kernel_size=3),
            nn.ReLU(inplace=True)
        )

    def forward(self, input):
        return self.conv(input)
```



conv1d

Parameters:

- **in_channels** (*int*) – Number of channels in the input image
- **out_channels** (*int*) – Number of channels produced by the convolution
- **kernel_size** (*int or tuple*) – Size of the convolving kernel
- **stride** (*int or tuple, optional*) – Stride of the convolution. Default: 1
- **padding** (*int, tuple or str, optional*) – Padding added to both sides of the input. Default: 0
- **padding_mode** (*str, optional*) – `'zeros'`, `'reflect'`, `'replicate'` or `'circular'`. Default: `'zeros'`
- **dilation** (*int or tuple, optional*) – Spacing between kernel elements. Default: 1
- **groups** (*int, optional*) – Number of blocked connections from input channels to output channels. Default: 1
- **bias** (*bool, optional*) – If `True`, adds a learnable bias to the output. Default: `True`

Shape:

- Input: (N, C_{in}, L_{in}) or (C_{in}, L_{in})
- Output: (N, C_{out}, L_{out}) or (C_{out}, L_{out}), where

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel\_size} - 1) - 1}{\text{stride}} + 1 \right\rfloor$$

dilation和stride值都为1时:

$$L_{out} = L_{in} + 2 \times \text{padding} - \text{kernel size} + 1$$

2. Model

3. self.pool11: maxpool1d

[64, 64, 2396] -> [64, 64, 798]

self.pool11=nn.Maxpool1d(3， stride=3)

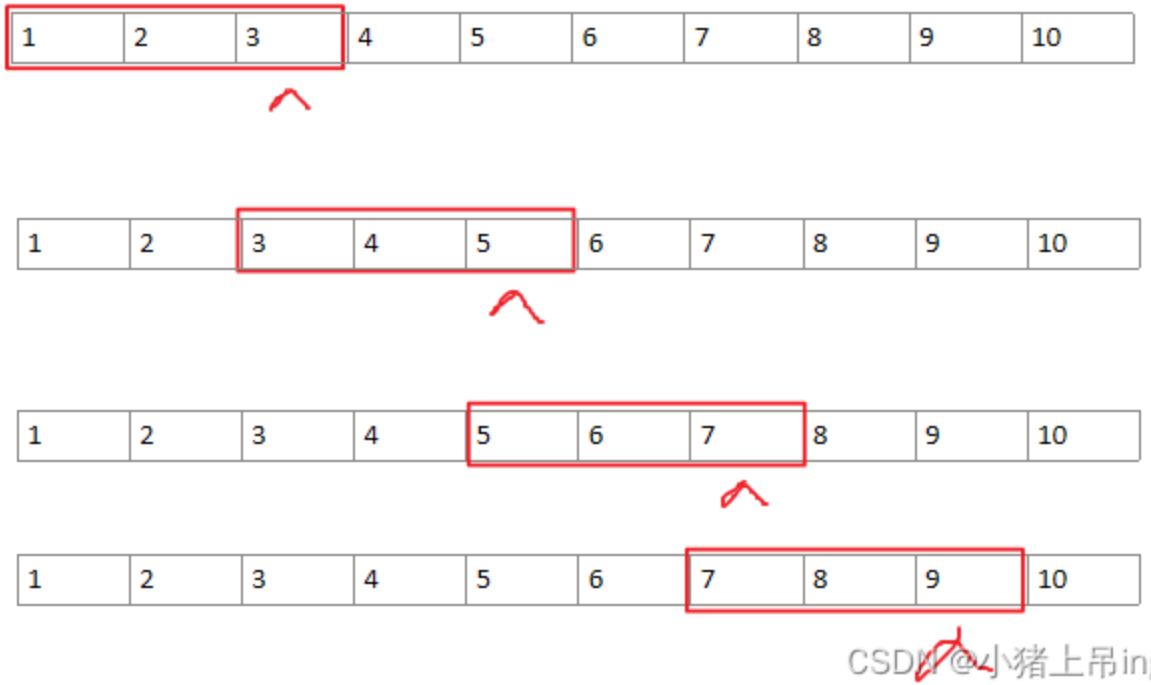
Parameters:

- **kernel_size** (*Union*[*int*, *Tuple*[*int*]]) – The size of the sliding window, must be > 0.
- **stride** (*Union*[*int*, *Tuple*[*int*]]) – The stride of the sliding window, must be > 0. Default value is `kernel_size`.
- **padding** (*Union*[*int*, *Tuple*[*int*]]) – Implicit negative infinity padding to be added on both sides, must be >= 0 and <= kernel_size / 2.
- **dilation** (*Union*[*int*, *Tuple*[*int*]]) – The stride between elements within a sliding window, must be > 0.
- **return_indices** (*bool*) – If `True`, will return the argmax along with the max values. Useful for `torch.nn.MaxUnpool1d` later
- **ceil_mode** (*bool*) – If `True`, will use *ceil* instead of *floor* to compute the output shape. This ensures that every element in the input tensor is covered by a sliding window.

Shape:

- Input: (N, C, L_{in}) or (C, L_{in}).
- Output: (N, C, L_{out}) or (C, L_{out}), where

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel\_size} - 1) - 1}{\text{stride}} + 1 \right\rfloor$$



2. Model

4. self.out: mlp

p15->Merge: [64, 512, 27] -> [64, 13824]
merge->output: [64, 13824] -> [64, 1]

```python
class MLP(nn.Module):
    def __init__(self, ch_in, ch_out):
        super(MLP, self).__init__()
        self.fc = nn.Sequential(
            nn.Linear(ch_in, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(inplace=True),
            nn.Linear(1024, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(inplace=True),
            nn.Linear(1024, 256),
            nn.BatchNorm1d(256),
            nn.ReLU(inplace=True),
            nn.Linear(256, ch_out),
        )

    def forward(self, input):
        return self.fc(input)
```

[64, 13824] -> [64, 1024]

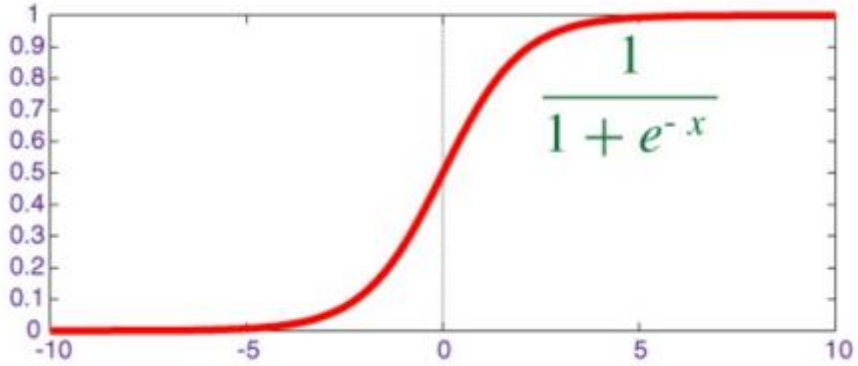
[64, 1024] -> [64, 1024]

[64, 1024] -> [64, 256]

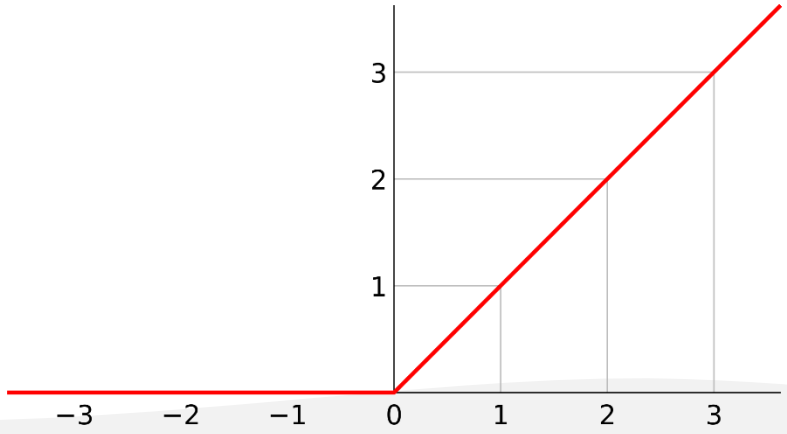
[64, 256] -> [64, 1]

5. F.sigmoid: map to (0,1)

在设定阈值为0.5情况，当输出结果大于0.5时，判定为正常；小于0.5时，判定为异常



Relu



3. Loss

Cross Entropy Loss

衡量估计模型与真实概率分布之间差异情况

对于单个样本，交叉熵损失的定义如下:

$$\text{CrossEntropyLoss} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

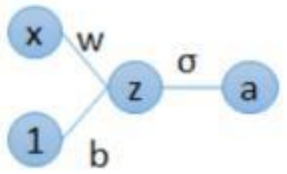
- C 是类别的数量。
- yi 是真实标签的一个one-hot编码。(若样本属于类别 i，则 yi = 1，否则 yi= 0)。
- $\hat{y}_i$是模型预测的第i类的概率

Example:

	Normal(1)	Abnormal(0)	CE Loss
Model 1	0.8	0.2	0.223
Model 2	0.6	0.4	0.511

$-1 * \ln 0.8$
$-1 * \ln 0.6$

假设给定输入为x，label为y，其中y的取值为0或者1，是一个分类问题。我们要训练一个最简单的Logistic Regression来学习一个函数f(x)使得它能较好的拟合label，如下图所示。



其中 $z(x) = w * x + b$， $a(z) = \sigma(z) = \frac{1}{1+e^{-z}}$ 。

交叉熵误差CEE（Cross Entropy Error）Loss

$$L_{cee} = -(y * ln(a) + (1-y) * ln(1-a))$$

$$\frac{\partial L_{cee}}{\partial w} = (-\frac{y}{a} + \frac{1-y}{1-a}) * \sigma'(z) * x$$

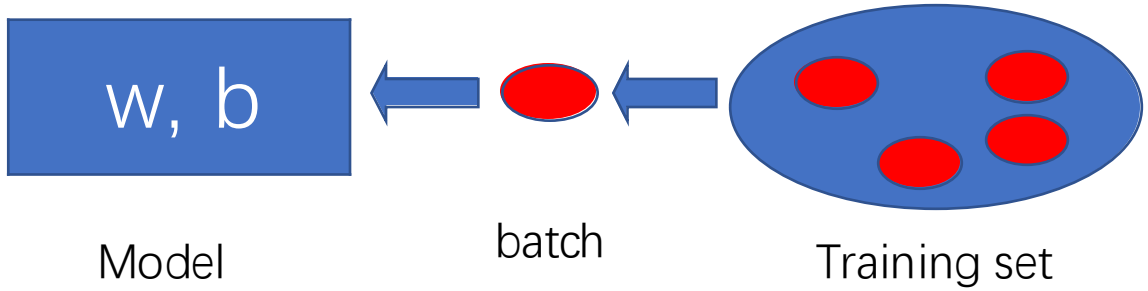
由于 $\sigma'(z) = \sigma(z) * (1-\sigma(z)) = a * (1-a)$，则:

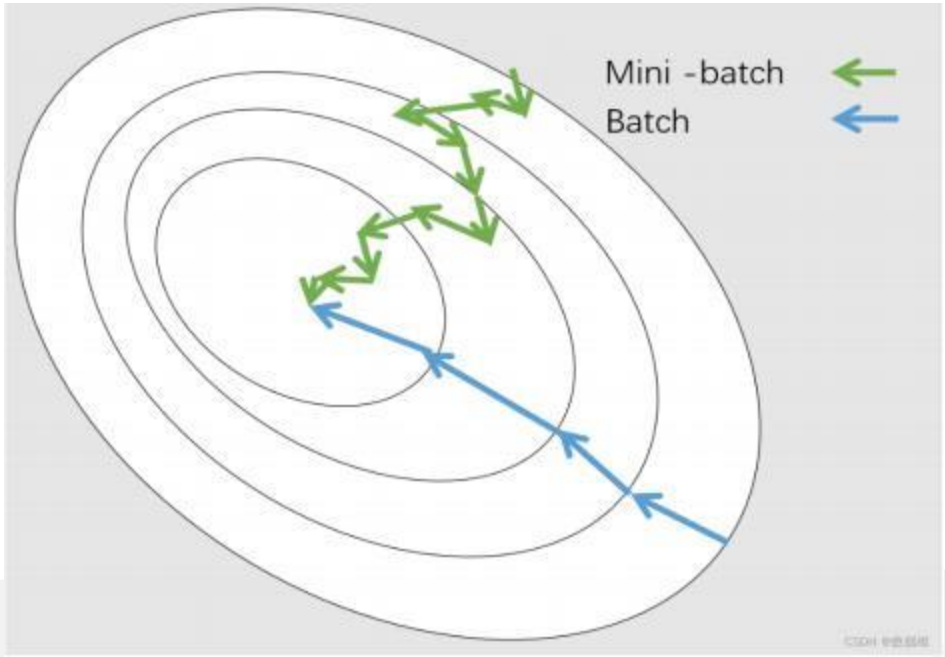
$$\frac{\partial L_{cee}}{\partial w} = (ay - y + a - ay) * x = (a-y) * x$$

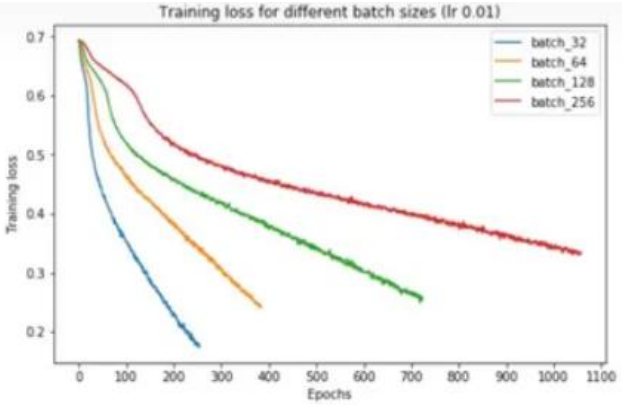
当真实y和预测a距离大时，梯度大，更新快。

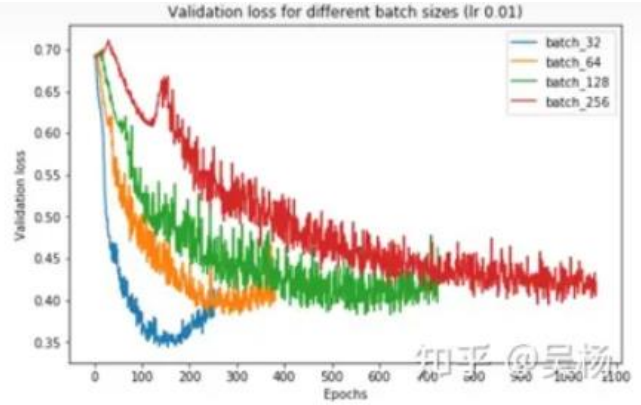
4. Hyperparameters

1. Batch_size：1,2,4,8,16,32,64,…

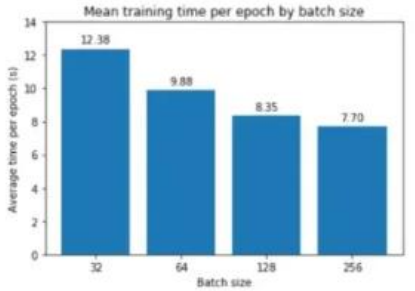


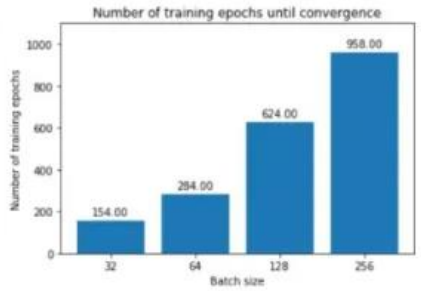






	Minimum training loss	Minimum validation loss
Batch size 32	0.174	0.344
Batch size 64	0.241	0.383
Batch size 128	0.250	0.383
Batch size 256	0.330	0.395





4. Hyperparameters

2.1 Learning rate (lr)

new_*weight* = *old*_weight - learning_rate * gradient

4. Hyperparameters

2.2 Learning rate schedule

前期 大力击打——大学习率

后期 轻轻击打——小学习率

图片来自:

https://www.cnblogs.com/peachtea/p/13532209.html

4. Hyperparameters

2.2 Learning rate schedule

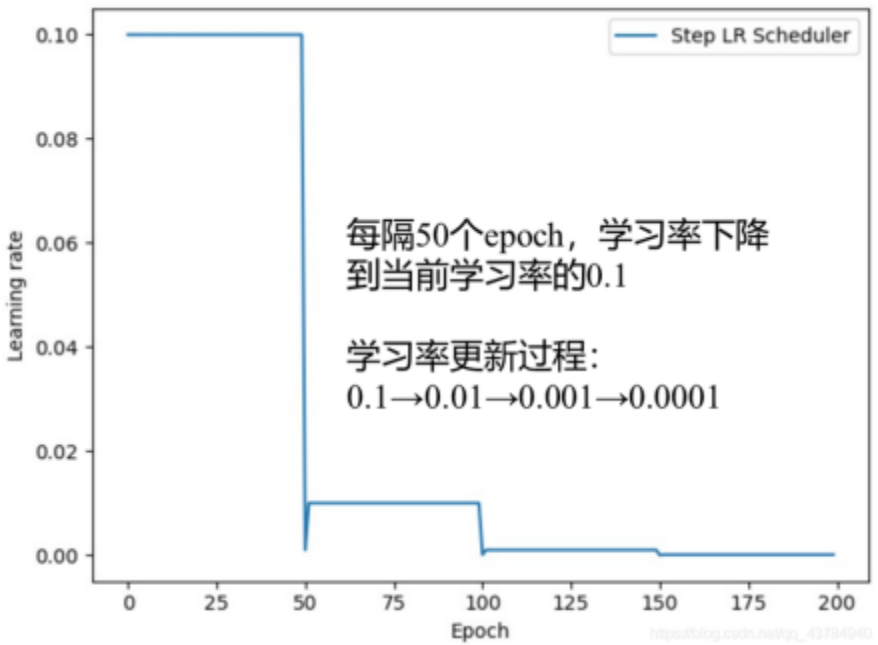
- StepLR:
  等间隔下降

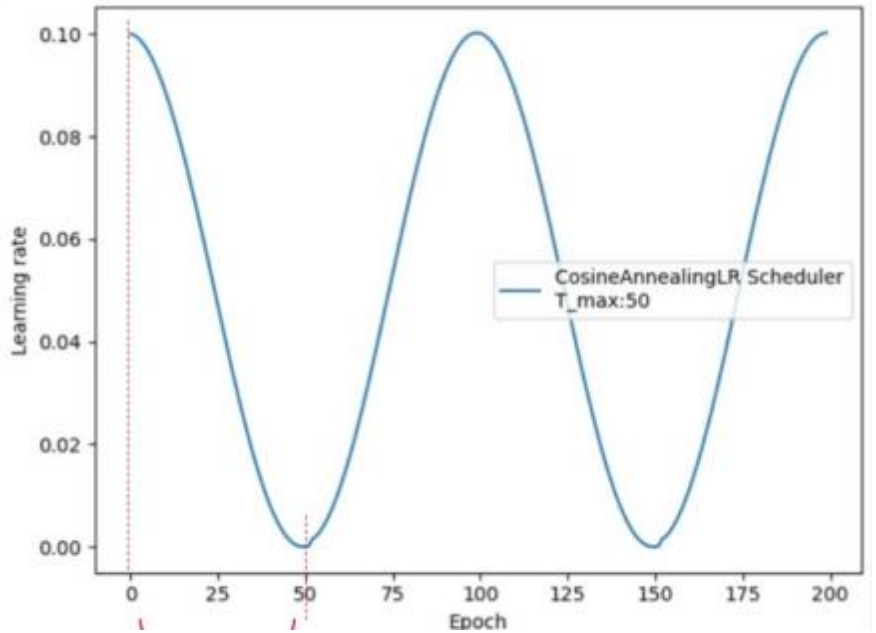
- MultiStepLR:
  自定义间隔下降

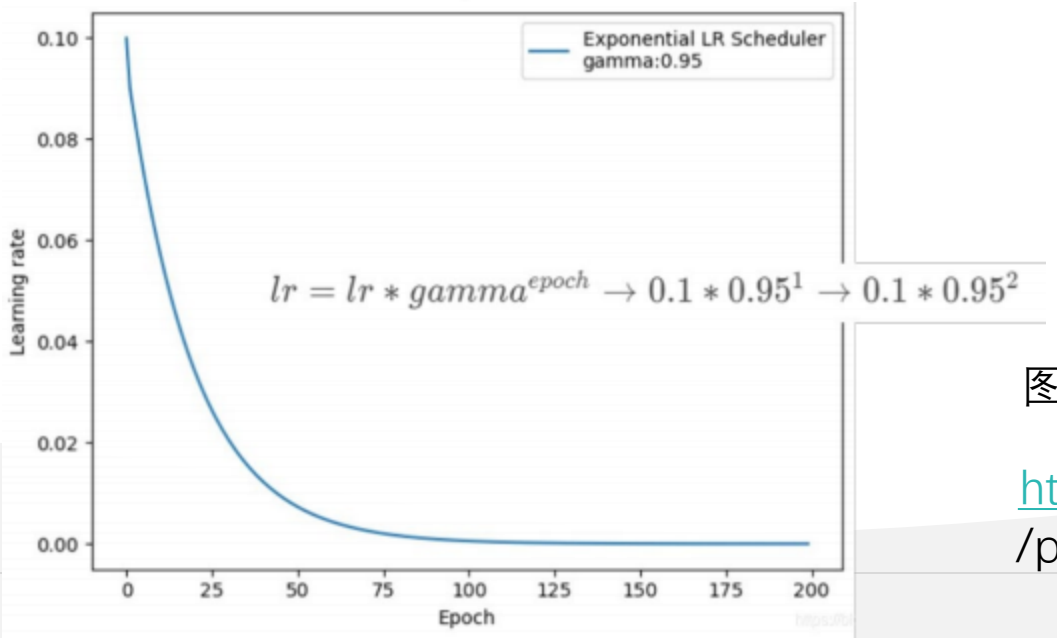
- ExponentialLR:
  指数下降

- CosineAnnealingLR:
  余弦周期变换

- ReduceLROnPlateau:
  当loss或指标不变时，自动调整







图片来自:

https://www.cnblogs.com/peachtea/p/13532209.html

4. Hyperparameters

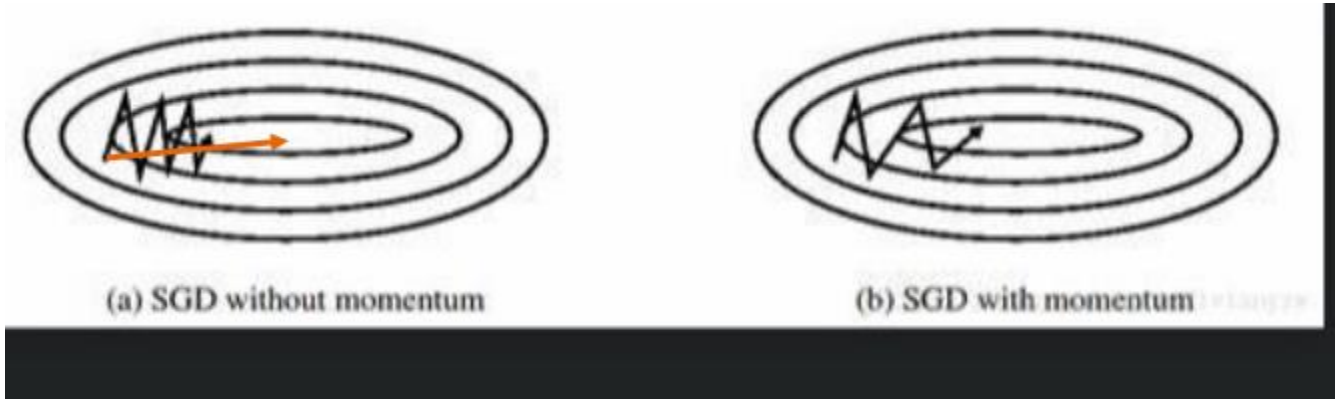
3. Optimizer

3.1 Stochastic Gradient Descent （SGD）

AdaGrad, Adam等等

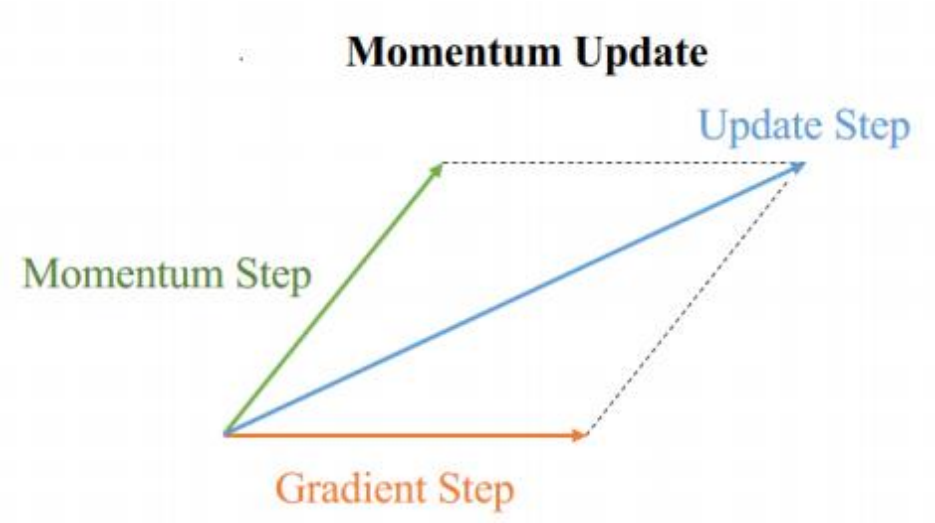
optimizer=torch.optim.SGD(model.parameters(), lr=1e-6)

https://blog.csdn.net/qq_44614524/article/details/114241259



3.2 SGD with momentum

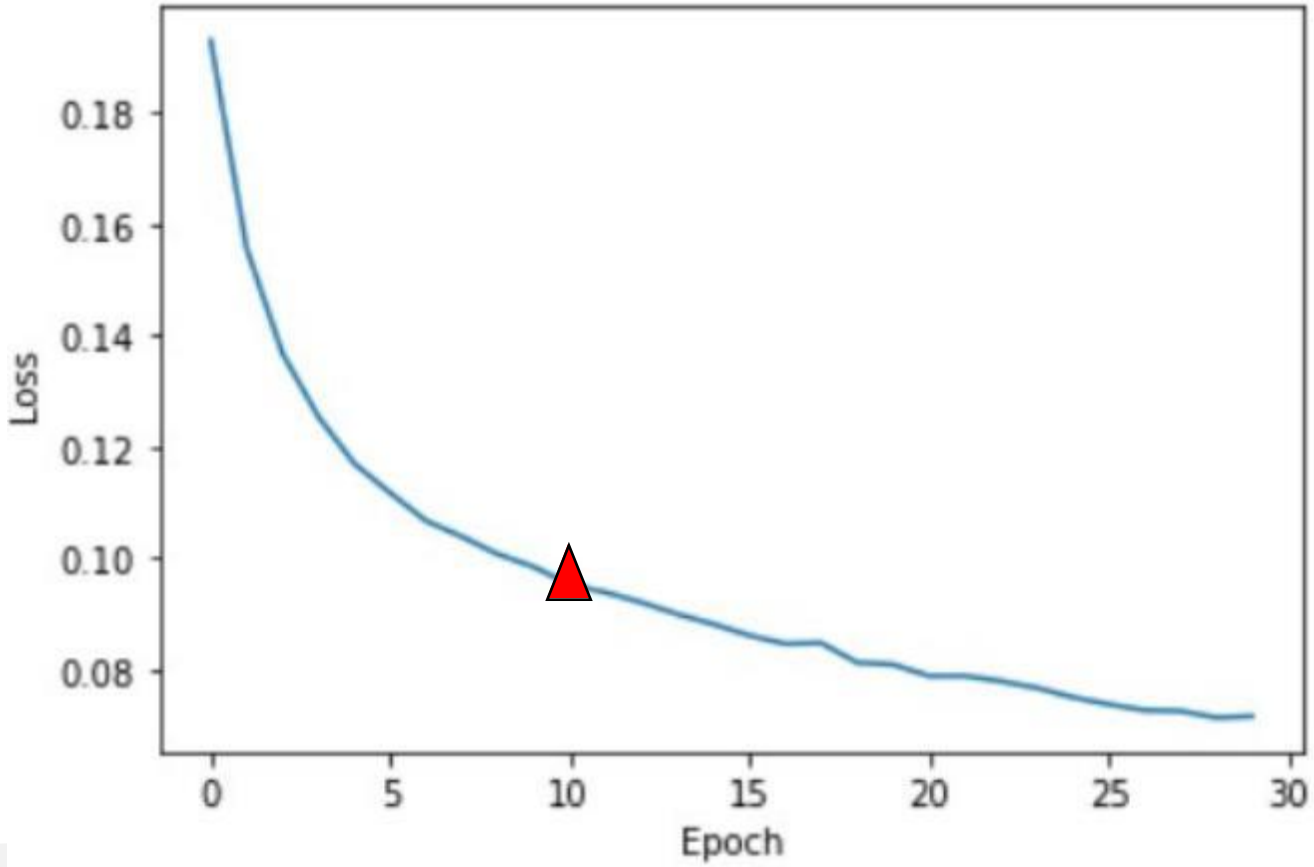
$$v_t = \gamma v_{t-1} + \eta g_t$$
$$\theta_{t+1} = \theta_t - v_t$$



4. Hyperparameters

4. Epoch: 训练完所有数据，称为一轮epoch

num_epochs=30



5. Training

```
# Start training !
for epoch in range(1, num_epochs + 1):
        print('Epoch {}/{}'.format(epoch, num_epochs))
        # Write your code here
        dt_size = len(dataloader.dataset)
        epoch_loss = 0
        step = 0
        process = tqdm(dataloader)
        for x, y in process:
            step += 1
            inputs = x.to(device)
            labels = y.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels.squeeze(2))
            loss.backward()
            optimizer.step()
            epoch_loss += loss.item()
            process.set_description(
                "epoch: %d, train_loss:%0.8f" % (epoch, epoch_loss / step)
            )
        epoch_loss /= step
         save_loss(10, epoch_loss)
# Save model
torch.save(model.state_dict(), 'weights10_%d.pth' % (epoch))
```

1. 加载数据，数据和模型要放在同一个设备上

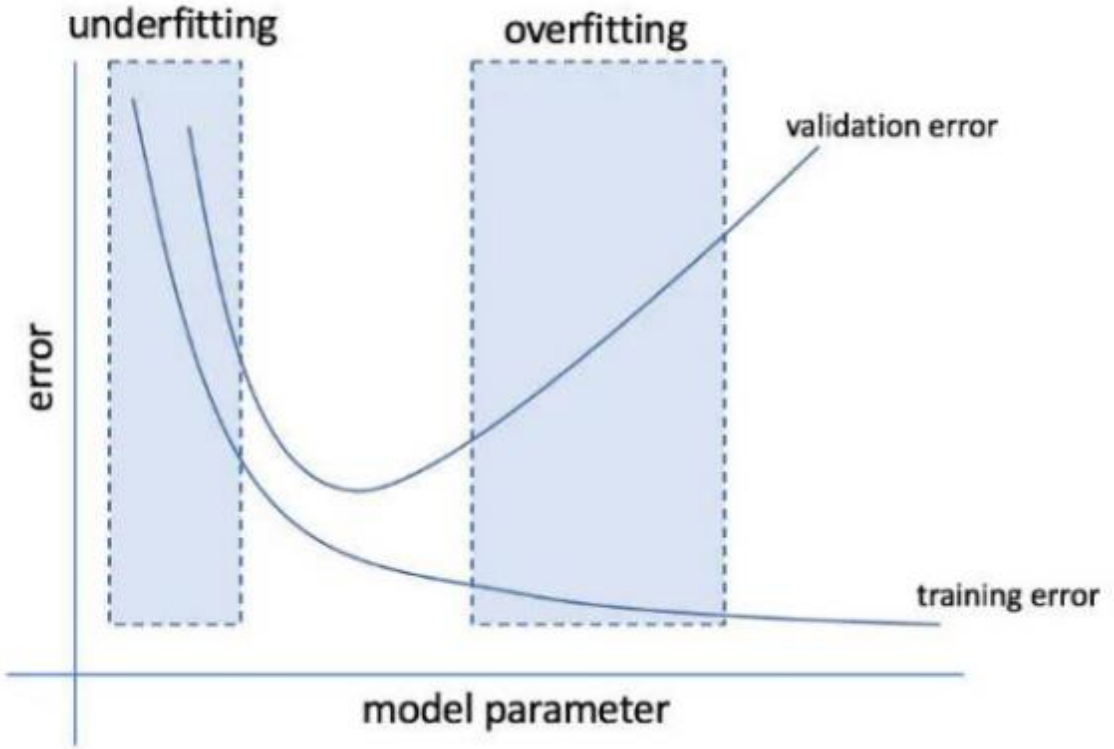
2. 优化器梯度置0，前向传播，计算损失函数

3. 反向传播，更新网络参数

5. Training

问题:

1. 只保留最后一个epoch结果，无法确定最优模型
2. 无法保证模型的泛化能力
3. 可能出现过拟合和欠拟合



5. Training

K-fold cross-validation:

Example: 5-fold cross-validation: