第二周

1. DataLoader与Dataset

1.1 DataLoader与Dataset

- 数据
  - 数据收集：img/label
  - 数据划分：train/valid/test
  - 数据读取：DataLoader
    - Sampler：生成索引index
    - DataSet：读取图片img和标签label
  - 数据预处理：transforms

DataLoader

```
# 构建可迭代的数据装载器
torch.utils.data.DataLoader()
DataLoader(dataset,     # Dataset类，决定数据从哪里读取
        batch_size=1,     # 批大小
        shuffle=False,     # 每个epoch是否乱序
        sampler=None,
        batch_sampler=None,
        num_workers=0,     # 是否多进程读取数据
        collate_fn=None,
        pin_memory=False,
        drop_last=False,     # 当样本数不能被batchsize整除时，是否舍弃最后一批数据
        timeout=0,
        worker_in it_fn=None,
        multiprocessing_context=None)
```

- `epoch`：所有训练样本都已输入到模型中，称为1个epoch
- `iteration`：一批样本输入到模型中，称为1个iteration
- `batchsize`：批大小，决定了一个epoch有多少个iteration

```
样本总数：87，batchsize：8
drop_last = True：1 epoch = 10 iteration     # 舍弃掉后七个样本
drop_last = Flase：1 epoch = 11 iteration
```

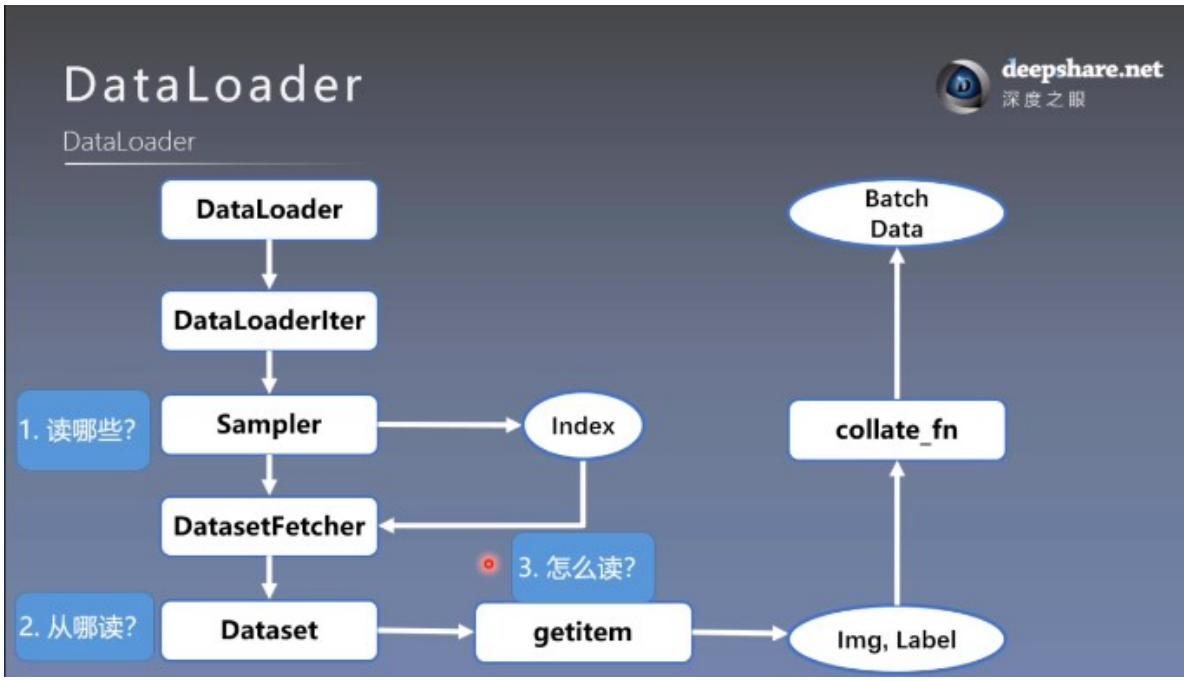
Dataset

```python
# Dataset抽象类，所有自定义的Dataset需要继承它，并且复写
torch.utils.data.Dataset()
class Dataset(object):
    def __getitem__(self, index):
        # 接收一个索引，返回一个样本
        raise NotImplementedError

    def __add__(self, other):
        return ConcatDataset([self, other])
```

1.2 人民币二分类

- 数据读取
  - 读哪些数据：Sampler输出Index
  - 从哪读数据：Dataset中的data_dir
  - 怎么读数据：Dataset中的getitem




```python
# TODO:代码示例
```

2. 图像预处理——transforms

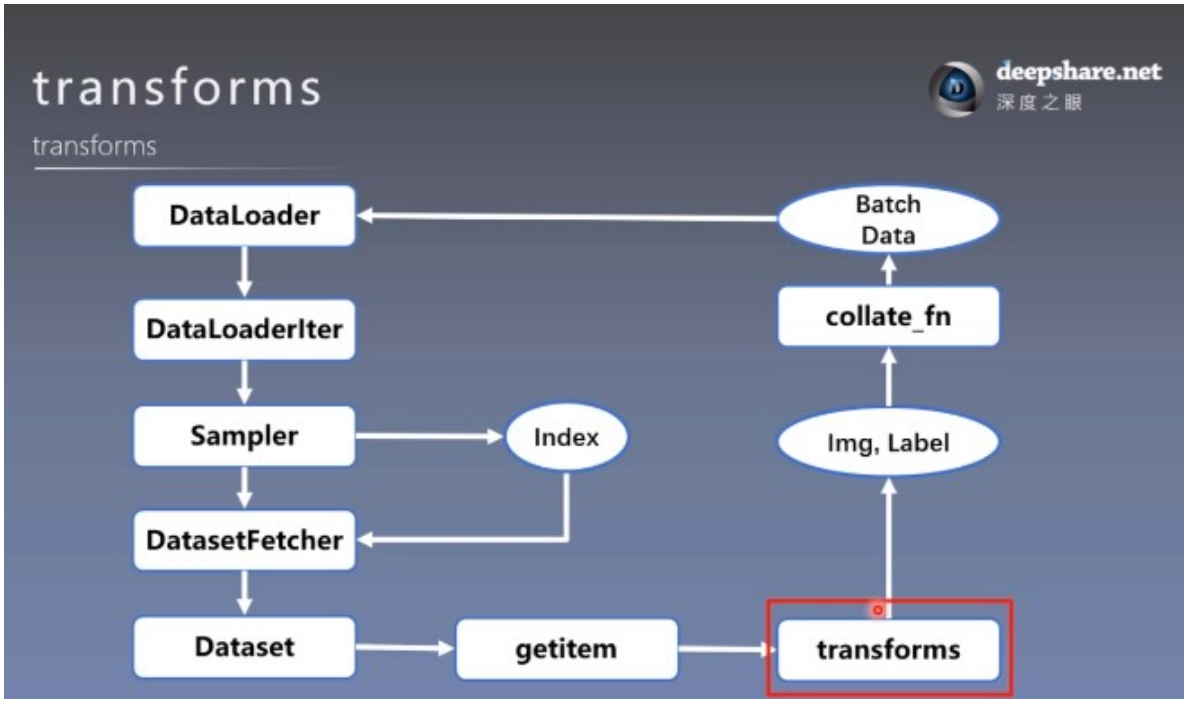
2.1 transforms运行机制

计算机视觉工具包：torchvision

- `torchvision.transforms`：常用的图像预处理方法
- `torchvision.datasets`：常用数据集，如MNIST/CIFAR-10/ImageNet
- `torchvision.model`：常用的模型预训练，如AlexNet/VGG/ResNet/GoogLeNet

常用图像预处理方法：transforms

- 数据中心化/标准化
- 缩放/裁剪/旋转/翻转/填充/噪声添加
- 灰度变换/线性变换/仿射变换/亮度、饱和度及对比度变换



2.2 数据标准化——transforms.normalize

- 加快模型的收敛速度

```
# 对图像的每个通道进行标准化
# output = (input - mean) / std
transforms.Normalize(mean,    # 各通道的均值
            std,    # 各通道的标准差
            inplace=False)    # 是否原地操作
```

2.3 作业

第二周作业1

3. 图像增强

3.1 数据增强

- 数据增强是对**训练集**进行变换，使训练集更丰富，从而让模型更具**泛化能力**，又称为数据增广/数据扩增
- 原则：使得训练集与测试集更接近
  - 空间位置：平移
  - 色彩：灰度图，色彩抖动
  - 形状：仿射变换
  - 上下文场景：遮挡，填充

3.2 裁剪

```
# 1. 从图像中心裁剪
transforms.CenterCrop(size)    # 所需裁剪图片的尺寸
# 2. 从图像中随机裁剪
# 当padding=a时，上下左右均填充a个像素
# 当padding=(a, b)时，上下填充b个像素，左右填充a个像素
# 当padding=(a, b, c, d)时，左/上/右/下分别填充a/b/c/d个像素
```

```
transforms.RandomCrop(size,     # 所需裁剪图片的尺寸
              padding=None,     # 设置填充大小
              pad_if_needed=False,     # 若图像小于设定size，则填充
              fill=0,     # 填充模式为constant时，设置填充的像素值，如(R,G,B)/(Gray)
              padding_mode='constant')     # 填充模式，共有4种
# constant：像素值由fill设定
# edge：像素值由图像边缘像素决定
# reflect：镜像填充，最后一个像素不镜像，例：[1,2,3,4] - [3,2,1,2,3,4,3,2]
# symmetric：镜像填充，最后一个像素镜像，例：[1,2,3,4] - [2,1,1,2,3,4,4,3]
# 3. 随机大小、长宽比裁剪图片
transforms.RandomResizedCrop(size,     # 所需裁剪图片的尺寸
              scale=(0.08, 1.0),     # 随机裁剪面积比例
              ratio=(3/4, 4/3)     # 随即长宽比
              interpolation)     # 插值方法，最近邻/双线性/双三次插值
# 4. 从图像上下左右及中心裁剪出尺寸为size的5张图像
transforms.FiveCrop(size)     # 返回一个tuple
# 5. 对上述5张图像进行水平或者垂直镜像获得10张图像
transforms.TenCrop(size,     # 所需裁剪图片尺寸
              vertical_flip=False)     # 是否垂直翻转
```

3.3 翻转与旋转

```
# 1. 翻转
# 水平翻转
transforms.RandomHorizontalFlip(p=0.5)     # 翻转概率
# 垂直翻转
transforms.RandomVerticalFlip(p=0.5)
# 2. 旋转
# 当degrees=a时，在(-a, a)之间选择旋转角度
# 当degrees=(a, b)时，在（a, b)之间选择旋转角度
transforms.RandomRotation(degrees,     # 旋转角度
              resample=False,     # 重采样方法
              expand=False,     # 是否扩大图片，以保持原图信息
              center=None)     # 旋转点设置，默认中心旋转
```

3.4 图像变换

```
# 1. 填充图像边缘
transforms.Pad(padding,     # 设置填充大小
              # 当padding=a时，上下左右均填充a个像素
              # 当padding=(a, b)时，上下填充b个像素，左右填充a个像素
              # 当padding=(a, b, c, d)时，左/上/右/下分别填充a/b/c/d个像素
              fill=0,     # 填充模式为constant时，设置填充的像素值，如(R,G,B)/(Gray)
              padding_mode='constant')     # 填充模式，
constant/edge/reflect/symmetric
# 2. 调整图像亮度/对比度/饱和度/色相
transforms.ColorJitter(brightness=0,     # 亮度调整因子
                  # 当brightness=a时，在[max(0, 1-a), 1+a]之间选择
                  # 当brightness=(a, b)时，在[a, b]之间选择
                  contrast=0,     # 对比度参数，同brightness
                  saturation=0,     # 饱和度参数，同brightness
                  hue=0)     # 色相参数
                  # 当hue=a时，在[-a, a]之间选择
                  # 当hue=(a, b)时，在[a, b]之间选择
# 3. 灰度图转换
transforms.Grayscale(num_output_channels)     # 输出通道数，1或3
```

```
transforms.RandomGrayscale(num_output_channels,
                p=0.1)    # 概率值
# 4.    仿射变换
# 仿射变换是二维的线性变换，由五种基本原子变换构成，旋转/平移/缩放/错切/翻转
transforms.RandomAffine(degrees,    # 设置旋转角度
                translate=None,    # 平移区间设置
                # 例：translate=(a, b)，a为宽，b为高
                # 图像在宽维度平移的区间为 -img_width * a < dx < img_width * a
                # 图像在高维度平移的区间为 -img_height * b < dy < img_height * b
                scale=None,    # 缩放比例
                shear=None,    # 错切角度设置
                # 当shear=a时，仅在x轴错切，错切角度在(-a, a)之间
                # 当shear=(a, b)时，则a设置x轴角度，b设置y轴角度
                # 当shear=(a, b, c, d)时，则a、b设置x轴角度，c、d设置y轴角度
                resample=False,    # 重采样方式，NEAREST/BILINEAR/BICUBIC
                fillcolor=0)    # 填充颜色设置
# 5. 随机遮挡
transforms.ToTensor()
transforms.RandomErasing(p=0.5,    # 执行遮挡的概率
                scale=(0.02, 0.33),    # 遮挡区域面积
                ratio=(0.3, 3.3)    # 遮挡区域长宽比
                value=0,    # 遮挡区域像素值，如(R,G,B)/(Gray)/'random'
                inplace=False)
# 6. 自定义lambda方法
transforms.Lambda(lambd)    # 匿名函数 lambda[arg1 [,arg2,...,argn]] : expression
```

3.5 transforms方法操作

```
# 1. 从多个操作中随机挑选一个
transforms.RandomChoice([transforms1, transforms2, transforms3])
# 2. 依概率执行一组操作
transforms.RandomApply([transforms1, transforms2, transforms3], p=0.5)
# 3. 打乱一组操作的顺序
transforms.RandomOrder([transforms1, transforms2, transforms3])
```

3.6 自定义transforms方法

```
# 自定义transforms要素：
# 1. 仅接收一个参数，返回一个参数
# 2. 注意上下游的输出与输入
class Compose(object):
    def __call__(self, img):
        for t in self.transform:
            img = t(img)
        return img
# 通过类实现多参数传入
class YoutTransforms(object):
    def __init__(self, ...):
        ...
    def __call__(self, img):
        ...
        return img
```

自定义方法——椒盐噪声

```
class AddPepperNoise(object):
    def __init__(self, snr, p):
        self.snr = snr
        self.p = p
    def __call__(self, img):
        # 添加椒盐噪声具体实现过程
        return img
```

3.7 作业

第二周作业2