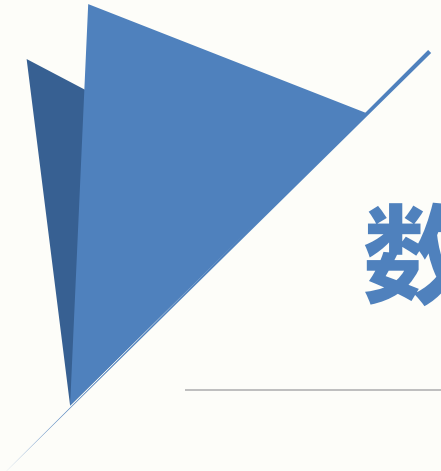
第04章 图像数据集和数据预处理

欧 新 宇

Thanks to LiMu-CS329 and LiFeifei-CS231n



数据准备（Data Preparation）

数据预处理概述

数据增广 (Data Augumentation)

主要作用是缓解过拟合问题，并增强模型的泛化能力。常见的数据增广方法包括水平和垂直翻转、2D旋转、缩放、随机裁剪、平移、对比度调整、色彩扰动、增加噪声、添加遮挡以及融合多种变换的混合变换等。

数据清洗 (Data Cleaning)

通过对噪声数据的识别，实现对坏数据（无法访问的数据）和脏数据（与任务无关的数据）的删除或标记或者对重复数据进行去除冗余操作等。

数据标注 (Data Annotations)

根据任务的不同，对数据进行类别、区域或像素的认知与识别，并生成标注数据列表。

数据集成 (Data Intergration)

实现多个数据集的融合，需要处理好标签的兼容与覆盖、数据的去重和二次标注，以及数据对齐等问题

数据规约 (Data Reduction)

又称基本数据预处理，主要包括像素归一化、均值消除、维度规约、数量规约、通道转换、数据类型变换等。

数据列表生成 (List Generation)

将数据集划分为训练、验证、测试和训练验证四个子集，并生成数据标签列表。分类任务列表的每一条数据都是[路径，标签]的二元组合，检测任务常见列表形式包括VOC格式（xml文件）和coco格式（json文件）。

数据清洗

数据样本常见的问题

样本无法读取

通常是损坏的样本，包括在网络下载过程中损坏或原本就是损坏的。这类样本会导致训练程序崩溃，可直接进行排除。

样本通道不一致

样本中同时存在彩色图像(depth=3)和灰度图像(depth=1)。这类样本需要将图像的维度进行统一。

命名不规范

一些程序设计语言对编码、字符和文件名长度都会有一些限制。不规范的命名方式或中文命名都有可能无法识别。这类样本可直接改名。

样本存在冗余

通常呈现为名称不同，但是内容、尺度、质量完全一致。这类样本需要删除冗余样本，以确保训练的平衡性和特殊性。

命名和内容不一致

样本的名称、描述和样本的实际内容不一致。这类样本通常需要进行描述修改或直接排除。

数据清洗

数据清洗的三种常用方法

01	02	03
修复数据	删除数据	索引数据
将无法使用的数据进行还原和修复或者对标注错误的样本进行重新标注。	直接将坏数据或脏数据从数据集中删除。该方法比较简单，但是会对原始数据集的统计信息造成改变。	对非法文件和坏文件进行索引和标注。利用索引文件可以在保留原始数据集不变的情况下，实现非法样本和坏样本的自动过滤和排除。

数据列表生成

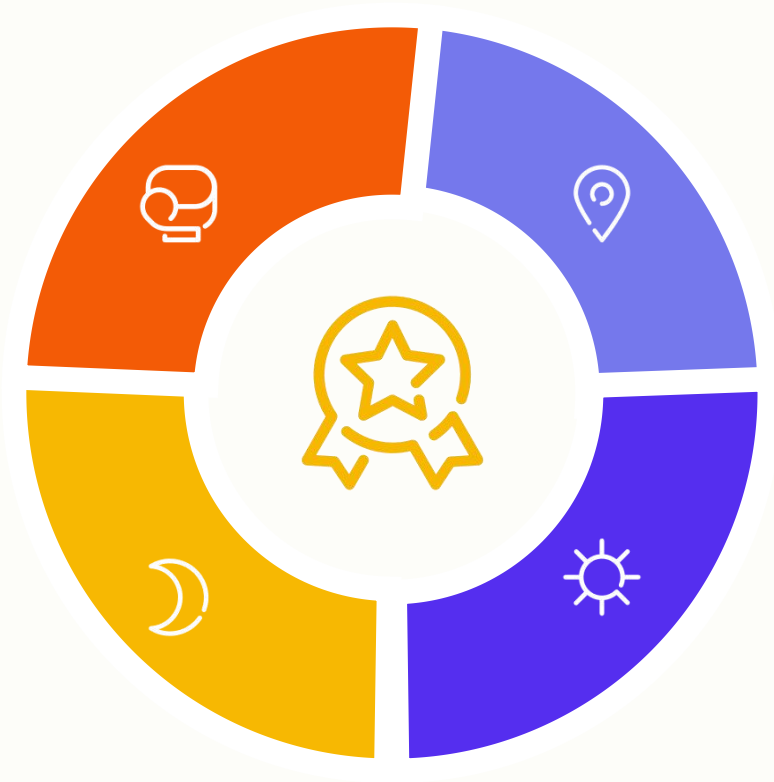
数据集的分类和功能

1 训练集 Training Set

用于对模型进行初步迭代训练，并输出调参模型。

2 验证集 Validation Set

用于在初步训练期间辅助训练，实现超参数的选择和评估。



3 训练验证集 Trainval Set

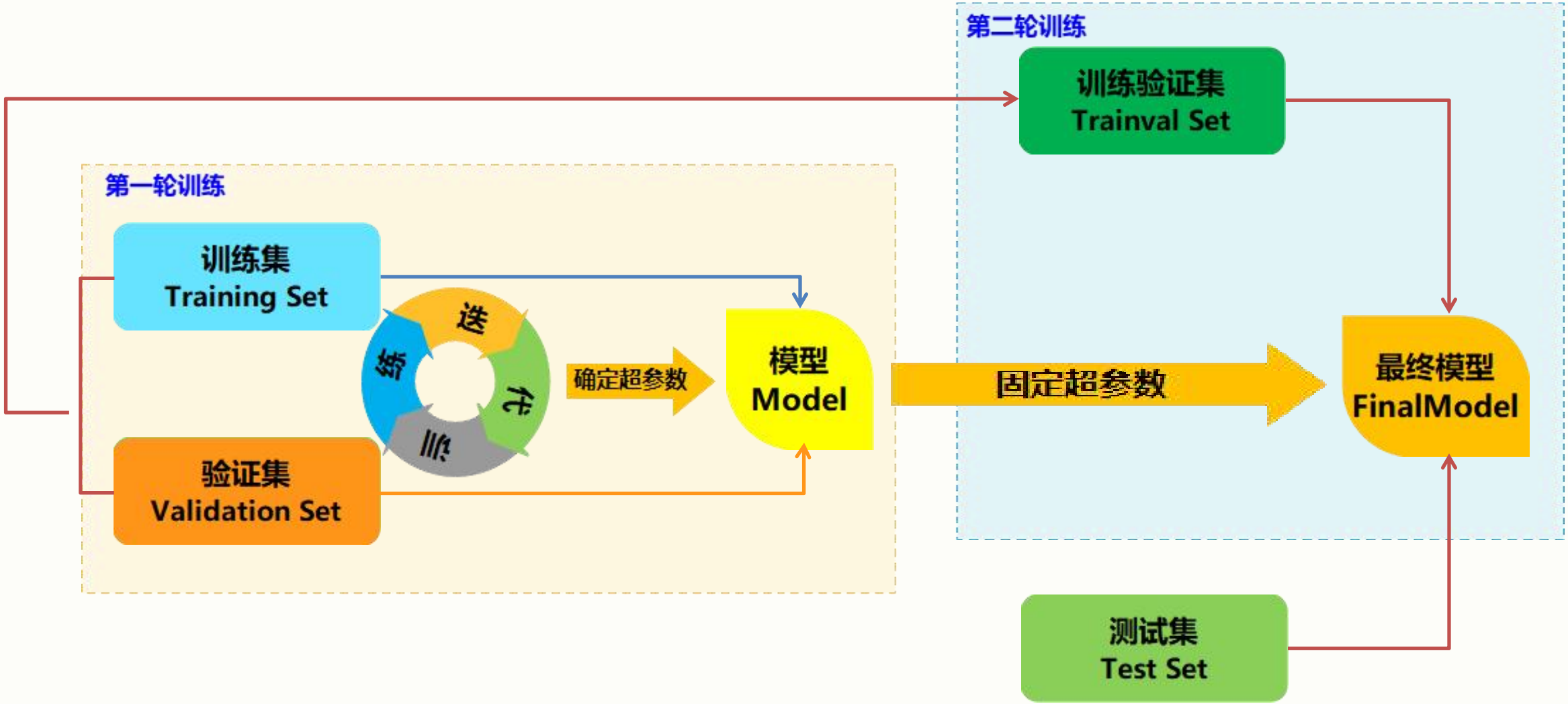
由训练集和验证集组合获得。用于在确定了超参数之后，对模型进行最终训练，并输出最终模型。

4 测试集 Test Set

用于最终模型训练完成后，进行最终模型的性能评估。

数据列表生成

深度学习训练过程中各个数据子集的使用方法



数据列表生成

基本流程

参数初始化

① 定义四个子集的样本数量、定义数据集信息字典

数据清洗

② 针对数据的问题进行针对性处理

路径定义

③ 数据集根路径、排除文件夹、数据列表文件路径、数据集信息文件、移除已存在的文件夹

数据划分

④
模式一：训练集+测试集
模式二：训练集+验证集+测试集+训练验证集

结果输出

⑤ 将统计信息存入数据集信息字典、输出字典json文件、print统计信息

数据列表生成

代码实现

参数初始化 ①

定义四个子集的样本数量、定义数据集信息字典

1. 载入必要的库
2. 初始化四个数据子集
3. 初始化类别数
4. 初始化数据集信息字典

```python
import os
import json
import codecs

# 初始化参数
num_trainval = 0
num_train = 0
num_val = 0
num_test = 0
class_dim = 0
dataset_info = {
    'dataset_name': '',
    'num_trainval': -1,
    'num_train': -1,
    'num_val': -1,
    'num_test': -1,
    'class_dim': -1,
    'label_dict': {}
}
```

数据列表生成

代码实现

```
# 本地运行时，需要修改数据集的名称和绝对路径，注意和文件夹名称一致
dataset_name = 'Zodiac'
dataset_path = 'D:\\Workspace\\ExpDatasets\\'
dataset_root_path = os.path.join(dataset_path, dataset_name)
excluded_folder = ['.DS_Store', '.ipynb_checkpoints']    # 被排除的文件夹
class_prefix = ['train', 'valid', 'test']
num_bad = 0
num_good = 0
num_folder = 0
```

数据清洗

② 针对数据的问题进行针对性处理

本例：实现坏样本排除

1. 定义根目录，并初始化坏样本和好样本的数量

```
img_path = os.path.join(dataset_root_path, prefix, class_name, image)
try: # 通过尝试读取并显示图像维度来判断样本是否损坏
    img = cv2.imread(img_path, 1)
    x = img.shape
    num_good += 1
    pass
except:
    bad_file = os.path.join(prefix, class_name, image)
    f_bad.write("{}\n".format(bad_file))
    num_bad += 1
```

2. 使用**试探策略**实现坏样本的探查。即如果无法正确读取或显示，则说明是坏样本。

3. 统计好/坏样本的数量

数据列表生成

代码实现

```python
# 2.1 数据集路径定义
dataset_name = 'Zodiac'                                          # 数据集名称（文件夹）
dataset_path = 'D:\\Workspace\\ExpDatasets\\'                    # 数据集根路径
dataset_root_path = os.path.join(dataset_path, dataset_name)
exclusion = ['.DS_Store', '.ipynb_checkpoints']                  # 定义被排除的文件

# 2.2 数据集列表路径定义
subPrefix = ['train', 'valid', 'test']                           # 定义数据集路径中的子文件夹，用于后续遍历，常见的分割方法如[train|val|test]，如
trainval_list = os.path.join(dataset_root_path, 'trainval.txt')  # 训练验证集列表路径
train_list = os.path.join(dataset_root_path, 'train.txt')        # 训练集列表路径
val_list = os.path.join(dataset_root_path, 'val.txt')            # 验证集列表路径
test_list = os.path.join(dataset_root_path, 'test.txt')          # 测试集列表路径
dataset_info_list = os.path.join(dataset_root_path, 'dataset_info.json')  # 数据集信息字典路径

# 2.3 检测数据集列表是否存在，如果存在则先删除。
if os.path.exists(trainval_list):
    os.remove(trainval_list)
if os.path.exists(train_list):
    os.remove(train_list)
if os.path.exists(val_list):
    os.remove(val_list)
if os.path.exists(test_list):
    os.remove(test_list)

# 2.4 读取数据清洗获得的坏样本列表，并将坏文件写入列表以备后续排除
bad_list = os.path.join(dataset_root_path, 'bad.txt')            # 设置坏文件列表路径
with codecs.open(bad_list, 'r', 'utf-8') as f_bad:               # 从坏文件中读取损坏的文件，并写入列表中
    bad_file = f_bad.read().splitlines()
exclusion = exclusion + bad_file                                 # 将排除文件与坏文件合并成一个文件
```

1. 定义根路径

2. 定义数据子集路径

3. 删除已存在的数据子集列表文件，防止冲突

4. 将损坏文件加入排除序列

路径定义

③ 数据集根路径、排除文件夹、数据列表文件路径、数据集信息文件、移除已存在的文件夹

数据列表生成

数据划分 ④

模式一：训练集+测试集

模式二：训练集+验证集+测试集+训练验证集

代码实现

1. 打开四个数据子集的列表文件

2. 判断扫描文件夹是否包含在排除文件夹内

3. 遍历所有样本，并按照已知的类别信息或求余数的方式匹配到不同的数据子集。并在分支语句中统计每个子集的数据量

```
# 遍历子文件夹，并分别输出四个不同的数据子集列表
with codecs.open(trainval_list, 'a', 'utf-8') as f_trainval, codecs.open(train_list, 'a', 'utf-8') as f_train, codecs.open(val_list, 'a', 'utf-8') as f
    for prefix in subPrefix:                                                    # 遍历子文件夹
        subDataset_dir = os.listdir(os.path.join(dataset_root_path, prefix))    # 获取子文件夹中的所有文件/文件夹列表
        for class_name_id in range(len(class_name_list)):                       # 遍历类别列表
            class_name = class_name_list[class_name_id]                         # 从类别名称列表中获取类别名称
            dataset_info['label_dict'][class_name_id] = class_name              # 将类别名称和对应的ID写入数据集字典的类别标签字典中
            images = os.listdir(os.path.join(dataset_root_path, prefix, class_name))  # 获取某个分类文件夹中的所有图片
            for image in images:
                if image not in exclusion:                                      # 判断图像文件是否是被排除的文件，若是则跳过
                    image_path = os.path.join(dataset_root_path, prefix, class_name, image)  # 获取图像的绝对路径
                    lable_id = class_name_id                                    # 获取图像的label_id
                    if os.path.join(prefix, class_name, image) not in exclusion:  # 判断文件是否是坏样本
                        if prefix == 'train':                                   # 如果文件是否在train文件夹中，若是则写入trainval和train文件
                            f_train.write('{}\t{}\n'.format(image_path, lable_id))
                            f_trainval.write('{}\t{}\n'.format(image_path, lable_id))
                            num_train += 1
                            num_trainval += 1
                        if prefix == 'valid':                                   # 如果文件是否在valid文件夹中，若是则写入trainval和val文件列
                            f_val.write('{}\t{}\n'.format(image_path, lable_id))
                            f_trainval.write('{}\t{}\n'.format(image_path, lable_id))
                            num_val += 1
                            num_trainval += 1
                        if prefix == 'test':                                    # 如果文件是否在test文件夹中，若是则写入test文件列表中
                            f_test.write('{}\t{}\n'.format(image_path, lable_id))
                            num_test += 1
```

数据列表生成

代码实现

```python
# 4.1 将数据集信息保存到json文件中供训练时使用
dataset_info['dataset_name'] = dataset_name
dataset_info['num_trainval'] = num_trainval
dataset_info['num_train'] = num_train
dataset_info['num_val'] = num_val
dataset_info['num_test'] = num_test
dataset_info['class_dim'] = len(class_name_list)

# 4.2 输出数据集信息到json文件
with codecs.open(dataset_info_list, 'w', encoding='utf-8') as f_dataset_info:
    json.dump(dataset_info, f_dataset_info, ensure_ascii=False, indent=4, separators=(',', ':')) # 格式化字典格式的

# 4.3 打印数据集信息。注意为方便可视化，我们使用display()替代print()对字典进行输出，但display()只支持notebook接口。
print('图像列表已生成，其中训练验证集样本{}，训练集样本{}个，验证集样本{}个，测试集样本{}个，共计{}个。'.format(num_trai
display(dataset_info)
```

1. 将数据子集的统计信息写入数据集信息字典中，并输出为json文件

2. 格式化字典显示样式

3. 输出统计数据，该数据同时保存到数据集信息字典中

结果输出

⑤ 将统计信息存入数据集信息字典、输出字典json文件、print统计信息

```
图像列表已生成，其中训练验证集样本7840，训练集样本7190个，验证集样本650个，测试集样本660个，共计8500个。
{'dataset_name': 'Zodiac',
 'num_trainval': 7840,
```

欧老师的联系方式

读万卷书 行万里路 只为最好的修炼

QQ：14777591 (宇宙骑士)

Email：ouxinyu@alumni.hust.edu.cn

Website: http://ouxinyu.cn

Tel：18687840023