**【图像分类】Pytorch多标签图像分类简明教程**

**简介**

基于image-level的弱监督图像语义分割大多数以传统分类网络作为基础，从分类网络中提取物体的位置信息，作为初始标注。

Pascal VOC2012的原始分割数据集仅包含1464个train图片和1449张val图片（共2913张），对于分类网络来说其数据量过小。而benchmark\_RELEASE分割数据集包括8498张train图片和2857张val图片（共11355张）。因此，许多论文中均选择使用benchmark\_RELEASE和原始Pascal VOC2012融合的增强数据集。

近期在复现论文过程中发现，使用增强数据集进行多标签分类时，某些图片缺少对应的标记，需要对照原始Pascal VOC2012数据集的标注方法，重新获取各类物体的标注信息，并完成多标签分类任务以及相应的指标评价。现将相关细节和部分代码进行解读，以帮助大家理解多标签分类的流程和相关注意事项。

**Pascal VOC2012原始数据集介绍**

Pascal VOC2012数据集包括五个文件夹：

1. Annotation：存放xml格式的标注信息
2. JPEGImages：存放所有图片，包括训练图片和测试图片
3. SegmentationClass：语义分割任务中用到的label图片
4. SegmentationObject： 实例分割任务用到的label图片
5. ImageSets：存放每一种任务对应的数据，其又划分为四个文件夹

* Action：存放人体动作的txt文件
* Layout：存放人体部位的txt文件
* Main：存放类别信息的txt文件
* Segmentation：存放分割训练的txt文件

本文是关于图片多标签分类任务的介绍，因此主要关注的为Annotation文件夹和ImageSets下的Main文件夹。

Main文件夹中包含了20类物体的训练、验证标签文件，其命名格式为class\_train.txt、class\_trainval.txt或*class*\_val.txt。

**benchmark\_RELEASE数据集介绍**

benchmark\_RELEASE数据集包括两个文件夹：

1. benchmark\_code\_RELEASE：相关评价指标的matlab文件
2. dataset：包括cls、img、inst三个文件夹和train.txt、val.txt两个文件

* cls：语义分割的mat标注文件
* img：分割图像
* inst：实例分割的mat标注文件

mat格式为matlab文件的一种，其中文件中主要包含了物体的类别、边界、分割标注三类信息，具体如下图所示：



**增强数据集介绍**

所谓增强数据集，共包含两个步骤：

* 将Pascal VOC2012和benchmark\_RELEASE两个数据集中的语义分割训练数据进行融合并剔除重复部分，即将"/benchmark\_RELEASE/dataset/"路径下的train.txt和val.txt文件与"/ImageSets/Segmentation/"路径下的train.txt和val.txt文件进行融合，获取最终的train.txt和val.txt文件，共12031个数据（8829+3202）。代码及注释如下（为了清晰展示步骤，将函数拆分，直接进行了书写）：

import os

from os.path import join as pjoin

import collections

import numpy as np# Pascal VOC2012路径

voc\_path = '/home/by/data/datasets/VOC/VOCdevkit/VOC2012/'

# benchmark\_RELEASE路径

sbd\_path = '/home/by/data/datasets/VOC/benchmark\_RELEASE/'

# 构建内置字典，用于存放train、val、trainval数据

files = collections.defaultdict(list)

# 填充files

for split in ["train", "val", "trainval"]:

# 获取原始txt文件

path = pjoin(voc\_path, "ImageSets/Segmentation", split + ".txt")

# 以元组形式打开文件

file\_list = tuple(open(path, "r"))

# rstrip清除换行符号/n，并构成列表

file\_list = [id\_.rstrip() for id\_ in file\_list]

# 不同阶段对应不同列表

files[split] = file\_list# benchmark\_RELEASE的train文件获取

path = pjoin(sbd\_path, "dataset/train.txt")

sbd\_train\_list = tuple(open(path, "r"))

sbd\_train\_list = [id\_.rstrip() for id\_ in sbd\_train\_list]

# benchmark\_RELEASE与Pascal VOC2012训练数据融合

train\_aug = files["train"] + sbd\_train\_list

# 清除重复数据

train\_aug = [train\_aug[i] for i in sorted(np.unique(train\_aug, return\_index=True)[1])]

# 获取最终train数据

files["train\_aug"] = train\_aug# benchmark\_RELEASE的train文件获取

path = pjoin(sbd\_path, "dataset/val.txt")

sbd\_val\_list = tuple(open(path, "r"))

sbd\_val\_list = [id\_.rstrip() for id\_ in sbd\_val\_list]

# benchmark\_RELEASE与Pascal VOC2012训练数据融合

val\_aug = files["val"] + sbd\_val\_list

# 清除重复数据

val\_aug = [val\_aug[i] for i in sorted(np.unique(val\_aug, return\_index=True)[1])]

# 清除val中与train数据重复的内容

set\_diff = set(val\_aug) - set(train\_aug)

files["train\_aug\_val"] = list(set\_diff)

* 同时==将"/benchmark\_RELEASE/dataset/cls"下mat格式的语义标签解析成图片，并与SegmentationClass文件夹下的图片进行融合。此部分代码可以参考该文件中的

[https://link.zhihu.com/?target=https%3A//github.com/meetshah1995/pytorch-semseg/blob/master/ptsemseg/loader/pascal\_voc\_loader.py](https://link.zhihu.com/?target=https://github.com/meetshah1995/pytorch-semseg/blob/master/ptsemseg/loader/pascal_voc_loader.py)

至此，增强数据集的train.txt、val.txt以及分割标注图片均已获得，可以愉快地用更大容量的数据集进行训练啦！

**标签文件制作**

系列一中我们介绍了Pascal VOC2012数据集的文件夹构成，在ImageSets/Main文件夹下包含了20类物体的标注文档，包括train、val和trainval三种划分。我们打开aeroplane\_train.txt文档可以看到，共有5717个训练数据，每个图像名称后面均对应了1或者-1，其中1表示图片中存在该类别的物体，-1则表示图片中不存在该类别的物体。增强数据集的train.txt和val.txt文件并没有各类别的标注信息，因此，我们需要仿照原有的格式，构建每个类别的标注文档。

Annotation文件夹下包含了所有图片标注信息的xml格式文件，其中<name>子项目下代表途中的类别信息。打开其中的一个xml文件我们可以看到，一个图中包含了多个类别信息，其中还有重复项，即图中存在相同类别的物体。我的思路是遍历train.txt和val.txt文档中每个图片对应的xml文件，获取其中的类别信息，然后判定类别信息是否包含当前类别，若包含则赋值1，反之赋值-1。对20个类别进行循环后，即可获得相应的标注文档。

接下来我将以训练标注文档的制作为展示，拆分步骤并结合代码进行详细的描述。

* 步骤1：读取train.txt文件获取训练图片

# 获取训练txt文件

def \_GetImageSet():

# txt路径

image\_set\_path = '/home/by/irn/voc12/train\_aug.txt'

with open(image\_set\_path, 'r') as f:

return [line.split()[0] for line in f.readlines()]

# 训练图片合集

img\_set = \_GetImageSet()

* 步骤2：读取对应的xml文件

# xml标注文件路径

annotation = '/home/by/data/datasets/VOC/VOCdevkit/VOC2012/Annotations'

# 构建xml列表

xml = []

for img in img\_set:

xml\_path = os.path.join(annotation,img + '.xml')

xml.append(xml\_path)

* 步骤3：根据xml中的<name>项，判定图片中是否存在该类别。读取<name>项之后，一定通过set()函数，清除其中的重复类别名称，否则会出现标签重复的情况

# 类别

\_VOC\_CLASSES = ['aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus',

'car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike',

'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor']

for x in xml:

# 获取每个name的地址

elem\_list = minidom.parse(x).getElementsByTagName('name')

name = []

# 读取每个地址的内容

for elem in elem\_list:

cat\_name = elem.firstChild.data

# 获取name

name.append(cat\_name)

# 删除重复标记

name = list(set(name))

# 根据类别写入标签文件

for cls in \_VOC\_CLASSES:

txt = '/home/by/data/datasets/gt/%s\_train.txt' % cls

if cls in name:

file\_write\_obj = open(txt, 'a')

gt = x[-15:-4] + ' ' +' '+ '1'

file\_write\_obj.writelines(gt)

file\_write\_obj.write('\n')

else:

file\_write\_obj = open(txt, 'a')

gt = x[-15:-4] + ' ' + '-1'

file\_write\_obj.writelines(gt)

file\_write\_obj.write('\n')

通过以上三个步骤，就可以生成train.txt在20个类别下的标注文档，效果如下图所示：

标签文件的制作是为了后续计算相应的评价指标，以更好的评价分类网络的性能。

**多标签矩阵的制作**

根据标签文件的制作，我们已经获取图片在每个类别下对应标签，如何将其转化成对应的矩阵形式，是我们的下一步工作。

在多标签分类任务中，我们可以构建一个1x20的矩阵作为图片的标签，其中对应的类别若存在，则置1，反之则置0。例如，如果图片中含有aeroplane和bicycle两个类别，其对应的标签矩阵应该为（1,1，0,0,0,0,0,0,0,0，0,0,0,0,0，0,0,0,0,0）。同样的，我们仍然可以根据xml文件信息，进行矩阵的搭建。

在本节中，我仍将通过步骤拆分，结合代码展示这一过程。

* 准备工作：设置文件夹名称，类别信息名称及其对应的数字

# 图片文件夹

IMG\_FOLDER\_NAME = "JPEGImages"

# 标签文件夹

ANNOT\_FOLDER\_NAME = "Annotations"

# 标签名称(不含背景)

CAT\_LIST = ['aeroplane', 'bicycle', 'bird', 'boat',

'bottle', 'bus', 'car', 'cat', 'chair',

'cow', 'diningtable', 'dog', 'horse',

'motorbike', 'person', 'pottedplant',

'sheep', 'sofa', 'train',

'tvmonitor']

# 标签转换为数字

CAT\_NAME\_TO\_NUM = dict(zip(CAT\_LIST,range(len(CAT\_LIST))))

* 步骤1：构建单张图片对应的标签矩阵

# 从xml文件中读取图片标签

def load\_image\_label\_from\_xml(img\_name, voc12\_root):

# 获取xml中的name项

el\_list = minidom.parse(os.path.join(voc12\_root, ANNOT\_FOLDER\_NAME,img\_name + '.xml')).getElementsByTagName('name')

# 构建标签空矩阵

multi\_cls\_lab = np.zeros((20), np.float32)

# 对xml中的name项进行操作

for el in el\_list:

# 读取name

cat\_name = el.firstChild.data

if cat\_name in CAT\_LIST:

# 转换为数字标签

cat\_num = CAT\_NAME\_TO\_NUM[cat\_name]

# 将标签矩阵中对应的位置赋1

multi\_cls\_lab[cat\_num] = 1.0

# 返回标签矩阵

return multi\_cls\_lab

* 步骤2：遍历所有的图片，生成对应的标签矩阵

# 从.txt文件中载入所有xml文件对应的标签

def load\_image\_label\_list\_from\_xml(img\_name\_list, voc12\_root):

# 返回所有标签矩阵

return [load\_image\_label\_from\_xml(img\_name, voc12\_root) for img\_name in img\_name\_list]

* 步骤3：生成含有所有标签矩阵的npy文件

# 加载图片list

def load\_img\_name\_list(dataset\_path):

# 获取.txt文件中的图片(含png和jpg,以及路径文件)

img\_gt\_name\_list = open(dataset\_path).read().splitlines()

# 读取图片名字

img\_name\_list = [img\_gt\_name.split(' ')[0][-15:-4] for img\_gt\_name in img\_gt\_name\_list]

# 返回值

return img\_name\_list

# 获取训练图片列表

img\_name\_list = load\_img\_name\_list(args.train\_list)

# 获取标签列表

label\_list = load\_image\_label\_list\_from\_xml(img\_name\_list, args.voc12\_root)

# 通过字典保存图片及其对应的标签

# 构建字典

d = dict()

for img\_name, label in zip(img\_name\_list, label\_list):

d[img\_name] = label

# 保存文件

np.save(args.out, d)

至此，所有的标签矩阵便构建完成了。

**评价指标计算**

多标签图像分类网络的性能需要根据平均准确率精度（mAP）来进行分析，而平均精度准确率均值需要先对每个类别的平均准确率进行计算。

根据分类网络我们可以得到图像在每个类别下对应的预测得分，其具体形式如下：

results =

{‘aeroplane’：{‘2007\_000032’:[0.7,0.8,......0.9],

......

'2011\_003276':[1.2,0.8,......0.3]}

......

'tvmonitor'：{‘2007\_000032’:[0.1,-0.8,......0.2],

......

'2011\_003276':[1.1,0.4,......0.8]}}

随后我们载入每个图像对应的类别标签，具体形式如下：

ground\_truth =

{‘aeroplane’：{‘2007\_000032’:[0,1,......0],

......

'2011\_003276':[1,0,......1]}

......

'tvmonitor'：{‘2007\_000032’:[1,0,......0],

......

'2011\_003276':[1,0,......1]}}

通过上述两个集合，我们可以分别计算每个类别的平均准确率，计算平均准确率的方法Pascal VOC官方已经给出，可以参照具体标准进行计算。具体代码如下：

# 每个类别的计算

def EvaluateClass(self, cls, cls\_results):

# 获取训练总数

num\_examples = len(self.image\_set)

# 构建gts

gts = np.ones(num\_examples) \* (-np.inf)

# 构建gts矩阵

for i, image\_id in enumerate(self.image\_set):

gts[i] = self.ground\_truths[cls][image\_id]

# 构建对应的confidences矩阵

confidences = np.ones(len(gts)) \* (-np.inf)

for i, image\_id in enumerate(self.image\_set):

confidences[i] = cls\_results[image\_id]

# 序号选择

sorted\_index = np.argsort(confidences)[::-1]

# 相应评价指标获取

true\_positives = gts[sorted\_index] > 0

false\_positives = gts[sorted\_index] < 0

true\_positives = np.cumsum(true\_positives)

false\_positives = np.cumsum(false\_positives)

recalls = true\_positives / np.sum(gts > 0)

eps = 1e-10

positives = false\_positives + true\_positives

precisions = true\_positives / (positives + (positives == 0.0) \* eps)

# 计算平均准确率

average\_precision = 0

# 根据Pascal VOC官方计算方法计算

for threshold in np.arange(0, 1.1, 0.1):

precisions\_at\_recall\_threshold = precisions[recalls >= threshold]

if precisions\_at\_recall\_threshold.size > 0:

max\_precision = np.max(precisions\_at\_recall\_threshold)

else:

max\_precision = 0

average\_precision = average\_precision + max\_precision / 11;

return average\_precision, list(precisions), list(recalls)

计算出每个类别的平均准确率后，则对所有类别的平均准确率求均值即可求得mAP值，在python代码中可以直接使用mean函数实现。

**训练**

在进行训练前需要注意一点，数据读取时我们需要同时获取图片名字、图片、标签三个信息，也是为了后续的评价指标计算做基础，这一点与传统单标签分类只读取图片和标签的方法不同，需要格外注意。

本文以Pytorch框架进行编写，进行了两种策略的训练方式

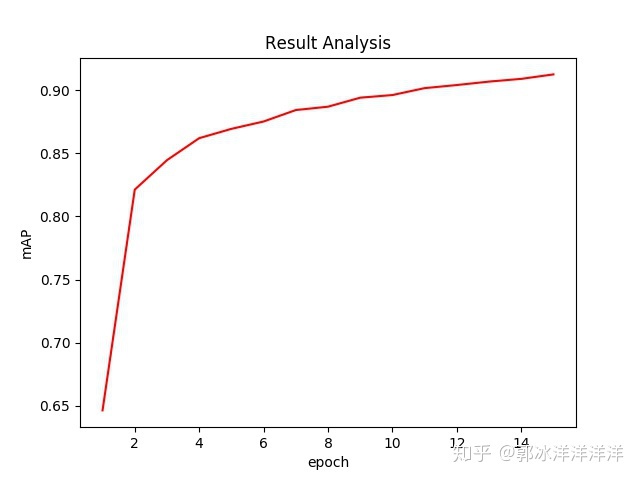
1、选择ModelA1作为训练网络（即resnet38），并使用对应的预训练数据，同时将全连接层转换为卷积层，学习率设置为0.01，batch\_size为4，损失函数选用hanming loss，采用SGD优化，在AMD 2600X + GTX 1070Ti搭建的平台，训练了约30个小时。

2、选择Resnet50作为训练网络，同时将全连接层转换为卷积层，学习率设置为0.01，batch\_size为16，损失函数选用hanming loss，采用SGD优化，在AMD 2600X + GTX 1070Ti搭建的平台，训练了约2个小时。

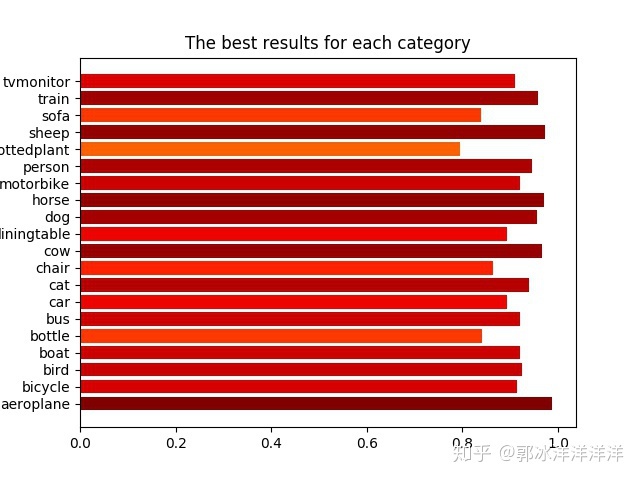
**结果**

通过训练我们发现，ModelA1取得的最优准确率为91.8%，Resnet50取得的最优准确率为90.3%，故此次结果分析暂时以ModelA1为准

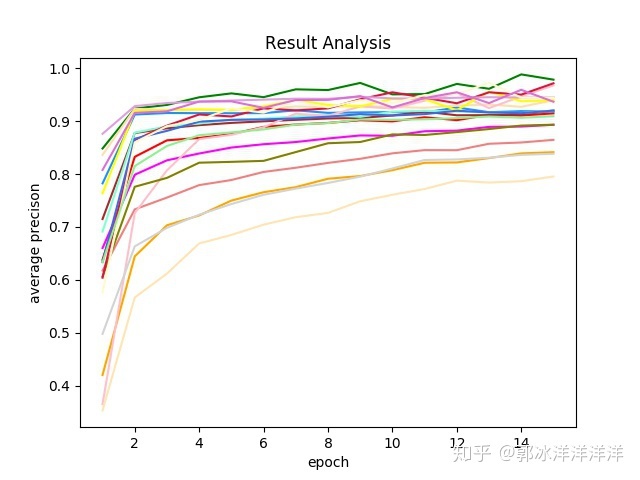
1、mAP



2、每个类别下的最优准确率



3、每个类别的平均准确率走势



以上就是整个多标签图像分类实战的过程，由于时间限制，本次实战并没有进行详细的调参工作，因此准确率还有一定的提升空间。