模式识别大作业——衣物颜色匹配

报告

班级： 自 93

学号： 2019010850

姓名： 王逸钦

邮箱： wangyiqi19@mails.tsinghua.edu.cn

完成日期： 2022/6/12

一　问题建模与形式化

我们希望达到如下目标：给定一组衣服图片和一组备选的描述颜色的中文 tag，算法可以给每个图片都分配一个 tag。

我们拥有如下数据：很多组图片，每组图片都已知备选 tag 以及每个图片被分配到了哪个 tag。

分析此问题，由于中文文本种类极多，且难以方便地找到描述颜色的中文语料，因此使用一个独立的深度学习模型处理文本是不现实的。另外由于多模态联合训练的两模态学习率难以很好地控制、配合，且本题的文本 tag 意思相对简单，所以不考虑多模态。

最终确定与“参考方案 1”类似的方法：word_to_label 映射表 + CNN + 逆映射算法

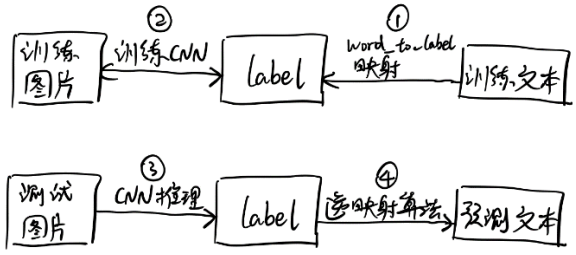

图 1 算法总流程

二　数据前处理

经过数据分析，我的 14 类样本（14 类样本如何生成后文详述）具有类别不均衡性，其中黑、白等常见色显著多于藏青、桔色等，详见图 2。(也可按 README.md 完成模型训练前的操作，然后运行 count_apperance.py 查看)。但训练集、测试集的 label 分布情况是很一致的。

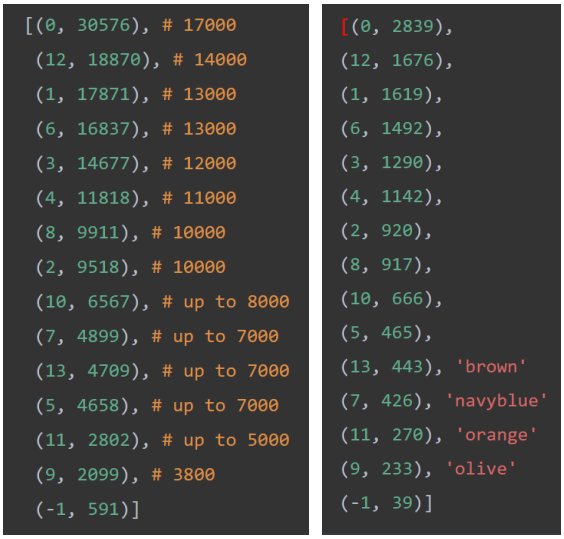

图 2 训练集、测试集的 label 分布情况(label, count)

由于二者分布一致，模型偏好频次较高的 label 也是可以接受的。但 CNN 训练并不鼓励出现过于悬殊的 lable 比例。权衡之下，我采取了两种方案并行的方式：一种方式完全不做处理；另一种方式将训练集进行一定的平衡，频次高的做采样，频次低的做复制，最终每类样本数量如图 2 的黄色注释所示（-1 为未能匹配 label 的文本，直接忽略）。

由于绝大部分图片已为 224*224 格式，裁切意义不大，且本任务颜色敏感，可能不适合进行通道归一化等操作，因此直接选择不进行 image augmentation。

由于我可以接入到一个拥有较大内存(264G)的训练服务器，因此我选择直接将所有图片数据转为 tensor 并一次性读入内存，这样在每次训练算法读取图片时不必进行磁盘 io 操作。由于我的算法在训练 CNN 时不考虑子“文件夹”的存在，因此可以丢弃这部分信息，直接将所有图片转为 3*224*224 后压成一个 tensor，所有 label 压成一个 tensor。

三　算法原理与实现过程

算法要点：word_to_label 映射表 + CNN + 逆映射算法

1) word_to_label 映射表

需要把总共 9000 多种的中文文本转化为颜色 label，才能用 CNN 进行学习。这步转化工作，我选择采用映射表方法。

经过对于所有文本的词频分析，我将颜色类别分为 14 类，如图 3 所示。

```
color2label = {
    'black': 0, 'white': 1, 'gray': 2, 'red': 3,
    'pink': 4, 'purple': 5, 'blue': 6, 'navyblue': 7,
    'green': 8, 'olive': 9, 'yellow': 10, 'orange': 11,
    'beige': 12, 'brown': 13,
}
```

图 3 颜色类别及其标号

建立一个 dictionary 词表，key 是 150 个左右本次数据中出现频次最高的中文词汇，value 是它们应该对应的颜色类别，词表由手工建立，如图 4 所示。对于给定文本，若其最后一个字是“色”则去掉“色”再查表；否则直接查表。

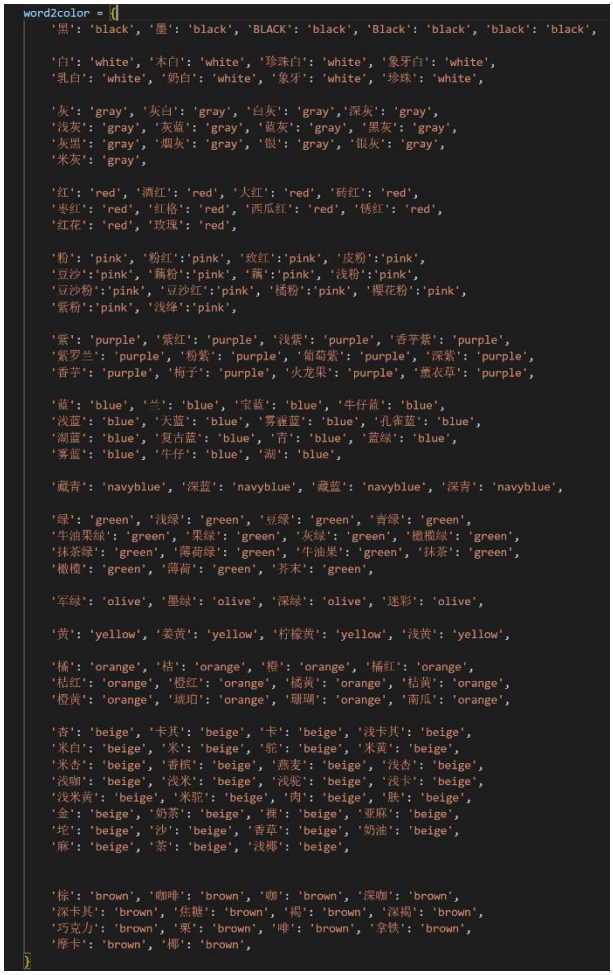

图 4 中文文本→颜色 label 映射表

即使建立了映射表，仍不可能覆盖全部 9000 多种中文文本。为此我设计了辅助算法——当文本不在映射表中时，从较大的子串到较小的子串、从后到前，依次查看子串是否在映射表中；若所有子串都不在映射表中，则返回 label=-1

例如“黑色裙子”不在映射表中，依次查询如下词汇“色裙子”、“黑色裙”、“裙子”、“色裙”、“黑色”，命中映射表，返回 label=0。

2) CNN

CNN 用于建立图像→label 映射关系。对于图像分类任务，我们有很多可用的经典网络结构，及其对应的在 imagenet 上训练得到的预训练权重。

在测试代码可行性阶段，我采用了未经预训练的 VGG11，最终效果平平但具备分类能力，说明算法在语法层面可行。（需要先完成【3）逆映射算法】才能测试，此处提前表述结果）

而后我尝试了预训练 ResNet18 模型，它参数量小，训练与推理速度快，最终在 SGD, momentum=0.9, weight_decay=1e-3, lr=1e-3, batch_size=96 的参数下，epoch19 的模型最优，达到测试 ACC=87.5%, EM=62.5%，获得一个 baseline。我又依次尝试了预训练的 ResNeXt50, ResNeSt50, DenseNet161 三个模型，其中 ResNeXt50 效果不佳，后两者效果较好，详见“结果对比”部分。此外，我还对比了数据集是否进行平衡对结果的影响。

3) 逆映射算法

对于每组图片，我们已知一个备选 tag 集合。当通过 CNN 得到每张图片的 14 维类别概率输出后，我们需要利用这一输出，结合备选 tag 集合，来确定给每个图片分配何种 tag。

由于每个 tag 都必定被使用，我决定采用先“tag 挑图”，剩余的图再“图挑 tag”。具体来说，先遍历每个 tag，选择对该 tag 预测概率最大的那张图，与该 tag 匹配，匹配后不再考虑该图。然后对“挑剩下”的图，再选择各个 tag 里面概率最大的，作为图的匹配 tag。

由此，每组图片都可以生成满足规则的预测 tag。

四 实验结果与分析

1) 有无 pretrain 对比

在算法-CNN 部分，“论证可行性”阶段，我先选择了未经过 pretrain 的 VGG11 模型，效果较差。这是因为我原本认为提取图像颜色信息并不太需要 imagenet 的权重，因为 imagenet 是物体分类。但改用模型能力相仿，但经过 pretrain 的 ResNet18 后效果大增，这也说明 pretrain 在具体任务中很有价值。

no pretrain, VGG11	ACC=80.8%	EM=41.8%
pretrained, ResNet18	ACC=87.5%	EM=62.5%

注：算力有限，我未能再进行控制模型类别的实验，
只有这组不同模型、不同 pretrain 状态的数据，仅作对比参考。

2) 优化器选择

有一个有趣的现象：pretrained ResNet18 在使用 AdamW with weight_decay 优化器时，模型精度仅可达到 70%左右便不再上升。但使用 SGD with momentum 就可以轻松达到 85%，加入 weight_decay 之后可以进一步提高至 87.5%。

鉴于 AdamW with weight_decay 融合了多种深度学习 tricks，其理应鲁棒性好、表现超过未经调优的 SGD。但经过我的实验在本任务上，对于 ResNet18 反而是 AdamW 效果很差，似乎难以解释。

在后续更大的模型训练中我仍然选择使用 AdamW，它们的性能表现都很正常，收敛速度也较快，符合对 AdamW 的预期。

3) CNN 选择 (Epoch 已挑选较优的) (均采用平衡的数据集)

Model	EpochNum	ACC	EM
VGG11(no pretrain)	80	80.8%	41.8%
ResNet18	**19**	**87.5%**	**62.5%**
ResNeXt50	24	81.5%	50.8%
ResNeSt50	**24**	**89.0%**	**67.8%**
DenseNet161	18	88.7%	65.9%

可见 2020 年提出的 ResNeSt50 确实可以比规模小且提出早的 ResNet18 获得一定的提升，但总的来说提升并不是跨越性的。其他与 ResNeSt50 同规模的模型表现不如它。

4) 是否平衡数据集

对比使用经过均衡、未经均衡数据的 ResNeSt50，结果如下：

Model	EpochNum	ACC	EM
数据均衡	24	89.0%	67.8%
未数据均衡	19	90.3%	70.3%

5) 我的最优方案

映射表 + CNN + 逆映射算法。CNN 采用未经均衡的数据集, Pretrained ResNeSt50, lr=1e-4, batch_size=256, num_epochs=20, AdamW, weight_decay=1e-3

达到 ACC=90.3%, EM=70.3%

五 被放弃的方案

在一开始，我想是否能通过基于回归的方法来解决这一问题。这是因为，颜色自带一些内生的特征空间，例如 RGB、HSV、HSL 等，其中视觉可分性最好的是 HSL。如果能将词表映射出的 14 种颜色类别，在 HSL 空间中的中心位置标定出来，把该坐标作为 CNN 的输出目标；在推理阶段，每一张图片也可以对应到 HSL 空间中的一个坐标，看该坐标与 14 个类别中心的距离，归到最近的一类即可。

这种思路的优势在于：颜色之间存在“远”和“近”的关系，例如我的标签里 red、pink 距离应该较近，blue、navyblue 距离应该较近，而 black、white 距离应该较远。当我们把这 14 个类的中心点建模在 HSL 空间中时，它们本身的距离就可以较好地反映它们语义上的距离，这听起来比直接贴标签分类更科学。

然而，这种思路也存在一个突出的问题。HSL 空间是圆柱体空间，在此空间根据 H(0°~360°)、S(0~100)、L(0~100)三维区间标定出的一个三维子空间形状可能很不规律。例如，我人眼观察 HSL 空间后对白色的标定范围是：H 任取，(S, L)在(0, 80), (100, 96)两点构成的直线上方。这是因为色相 H 对白色并不关键，亮度 L 高就可以；但当色彩饱和度 S 低时亮度稍低也可以被判定为白色，当 S 高时 L 必须贴近 100 才可被判定为白色。可以发现，人类视觉所判定的每个类别，其在 HSL 中的子空间形状可能很不规则，对于每一维度变化的敏感程度也相互不同，这给在 HSL 空间中定义“距离”带来了极大的挑战，这个空间距离很难定义得接近“语义距离”。而且过多的人为干预，也给最终的结果带来了很大的不确定性。

最终，我没有选择这种基于回归的方案。

六 亮点、困难与解决方案、收获

1) 亮点

- 思路清晰，词表处理方法可以命中绝大部分文本，逆推理算法具备有效性
- CNN 使用性能较优的 ResNeSt50

2) 收获

- 之前 DeepLearning 相关经历都是图片→标签，这是我首次处理文本数据。
- 本问题引导我们思考如何有效利用组内部的信息，且数据质量很高，是一个性能上限高的问题。截至完成时间(2022-06-12)，我的 ACC=90.3%, EM=70.3%暂列排名第 7，后续应会掉出排行榜。

附：参考资料

train_model 函数参考李沐-《动手学机器学习》课程示例代码：
https://zh-v2.d2l.ai/d2l-zh.zip

附：README.md

衣物颜色匹配大作业 自93王逸钦 2019010850

文件列表：

Dir	FileName	Description
code	generate_json.py	运行以生成json文件夹和两个辅助json文件
	process_balance_data.ipynb	运行以生成tensor格式的数据
	resnet18.py	运行以训练ResNet18模型
	resnest50.py **BEST**	运行以训练ResNeSt50模型
	densenet161.py	运行以训练DenseNet161模型
	inference.ipynb	运行以推理json结果，存于本目录json文件夹
	dataset.py	内定义DataSet类
	get_label.py	内定义get_label函数，输入中文词输出标号
	train_model.py	内定义train_model函数
	count_appearance.py	辅助函数，用于分析数据集，对训练不必需
	my_test_data_ResNeSt50....json	最优的json结果，对训练不必需
	README.md	
data	medium文件夹	需要预先放置medium数据
model (空目录)		放置生成的模型
tensor_data(空目录)		放置生成的tensor格式的数据

我的训练环境：

Ubuntu 16.04.7 LTS，264G ram，4*1080Ti

python3.7.11, torch 1.8.1+cu101, torchvision 0.9.1+cu101

nvidia driver version: 455.23.05，CUDA Version 11.1

使用方法：

- 按以下步骤可进行数据处理、模型训练、推理生成结果
- 注意，本项目采用直接把所有图片一次性load的方式，请保证运行内存不低于128G
- 注意，本项目直接存储原始model，且使用多GPU训练，需保证训练时的可用显卡编号和推理时一致

1. 按照上表所示，建立出所有的空目录
2. cd到code目录下，运行generate_json.py
   这将在code目录下生成json文件夹(内含有color2label.json, word2color.json)
3. 运行process_balance_data.ipynb，依次产生未经平衡和经平衡的tensor格式数据，存于tensor_data目录 (耗时1h)
4. 选择一种模型，运行该文件，每个epoch的模型均会存储在model目录。若使用ResNeSt50，请先安装：

```
pip install resnest --pre
```

5. 修改inference.ipynb中的参数（指定模型的路径），运行，推理结果保存在code文件夹下的json目录中

直接下载模型最优模型 (该模型推理时需要GPU0,1,2,3均可用)：https://cloud.tsinghua.edu.cn/f/d0fcd092a9024ba286e8/