2022年春季《模式识别与机器学习》大作业报告

姓名：郑智骋

一、 问题形式化与切入点分析

本次模式识别大作业要求我们将某购物平台的一系列同种版型服饰的不同颜色款式实物图片与给定的一系列颜色中文标签进行匹配并输出对应结果。

1.1 该问题形式化表述：对于可变的 $m > n$，

输入 1：{服饰款式图片 0，服饰款式图片 1，…，服饰款式图片m}

输入 2：{色彩文本标签 0，色彩文本标签 1，…，色彩文本标签n}

预期输出：对于$i_0 \sim i_m \in [0, n]$，

{服饰款式图片 0：色彩文本标签i_0}；

{服饰款式图片 1：色彩文本标签i_1}；

…

{服饰款式图片 m：色彩文本标签i_m}；

1.2 该问题的分析

从笔者的角度，该问题看似是一个分类问题，但由于分类的标签也是一个可变的变量，所以本题的本质是一个将两种模态的数据（文本+图像）进行特征提取并使二者的特征空间匹配的问题。而分类则是促使两模态数据学习到良好表征的一种重要手段。因此，针对该任务，我们首先要分别对文本和图像分析，如何来抽提它们中有效的特征并使其适合于特征匹配，所谓的特征具体到本任务就是色彩。

首先是文本，本任务中文本所体现的色彩特征是由字或者词反映的。由于不涉及句子与篇章的处理，所以几乎不需要考虑语序、上下文等的问题。因此文本的处理流程设计思路比较直接，主要采用传统的 Embedding 方法：1. 先将训练数据与测试数据（注：由于这里并没有用到测试集的标签，所以不存在测试集数据泄露的问题，可以使用测试集的数据）文本中的词与字都存储为字符到序号的字典（保存为文件），再训练一个字词序号到 Embedding 的 Embedding 层；2. 如

果还要考虑上下文关系（如“黑色上衣搭配淡黄长裙”），则再加入 LSTM、RNN 等循环网络层进行处理。

其次是图像，经过实例的观察，要识别本任务中的图像色彩且与标签写的色彩匹配上，是比较困难的。因为标签不一定会对整个服装的色彩进行描述，而只会描述一套不同色彩的服装之间具有差异的那部分色彩，所以需要模型具备对比不同图像之间的色彩的能力。笔者起初希望能训练一个可以将全部不同色彩图片都同时纳入决策范围的模型，但是由于图像数目是变化的，所以不可能有一套固定结构的网络来处理该问题，如果要对比变化数目的图像特征，可能需要循环网络或者注意力层。对于这一问题，笔者考虑了不同的解决方案，譬如引入图像特征之间进行对比的循环网络，设计对比损失函数来增大非同类图像之间特征差异等等方案。但经过大量的实验，最后还是选择了用一个模型来抽提单张图像中的特征，通过对比损失函数来尽可能地增大非同类图像特征之间的差异，而鼓励同类图像特征之间趋于更加相似化，取得了笔者训练模型中的最好效果。

因此，笔者的想法是，即使模型不能够观测到其他图片的色彩分布情况，依旧能够通过设计合适的 Loss 函数，来引导模型学习到图片的合理色彩表征。也就是说，通过模型处理单张图片而非 Group 中的全部照片，也能够得到图片中的各种色彩分布，而总有一些主要色彩的特征是在别的色彩的图片特征中不曾出现的，而同类色彩的图片的色彩分布又会比较接近，所以暂且先不管标签如何，先鉴别出两图片是否是同色彩的图片，再考虑标签匹配会是一个不错的想法。

于是就引出了“先聚类，再匹配标签”的思路。该思路的流程是：

1. 训练文本-图像联合模型。先训练模型使其能够抽取出单张照片中受关注主体的色彩分布（一般使用 ImageNet 预训练模型是能够识别出哪些是图片中的主体的，在本任务中即为服饰或模特）；

2. 聚类。训练得到模型之后，设计一个聚类函数，在预测时，将一个 Group 中的图片的特征进行 k 聚类（k 为标签个数），本实验中用 Rough K-Means 聚类并得到每一张图片对每一个标签（称为虚拟标签，因为此时未一一对应）的隶属度函数，于是就可得到每张照片的虚拟标签；

3. 对虚拟标签进行匹配。我们假设训练后的文本-图像联合模型的文本 Embedding 与图像 Embedding 的特征空间是基本重合的。所以对聚类迭代收敛的 k 个聚类中心的 Embedding 与 k 个文字标签的 Embedding 进行相似度计算，得到相似度矩阵，于是可以设计算法选出虚拟标签的真实值。最终就得到了可以输出的结果。

以上基本就是本实验最佳结果所用的算法梗概，下面深入分析流程与算法。

二、 数据处理流程

整个项目从数据集到输出 json 预测结果的全过程处理流程及其代码文件对应关系如下：

2.1 安装依赖库、修改代码中的数据集地址等准备工作

2.2 注册文本数据 token DATA_REGISTER.py

提取出原始数据集的文本标签（训练集+测试集）中的所有可能的词语+单字，生成一个字符（串）映射到 token/序号的字典。（代码中也包含了生成 token 到字符串的字典与字符出现的频数字典，项目中其实并没有用到）

会输出一个 txt 文件在同目录下的/words 文件夹中。

2.3 训练模型 TOP_VIEW_VIT_VERSION8.py

原数据预处理

数据预处理相关的类与函数在 **data_loading.py** 中

① Taobao_Tokenizer 类。将文本数据用 jieba 库（精准模式）来辅助分词，并查表转化为 token 输出

② Taobao_Dataset 类，对图像、文本联合处理并输出。由于每一组数据的规模不固定，所以一次只能输出一组商品的文本、图像数据，即 batch_size=1。首先遍历文件夹中的文件，识别出 json 与 jpg 文件，读取 json 文件中的字典。分别处理图像、文字。

a) 对图像：使用预定义的 transform 对象进行变换并张量化，将所有图片 tensor 堆叠成一个总的张量输出。具体处理过程在后文将详细介绍。

b) 对文字：首先对标签中的字符串进行 jieba 分词，分别用 jieba 中的搜索引擎模式与全切分模式整合出所有可能的词语及其组合，再加上所有单个字，构成一个无重复集合 words_set。然后计算出所有色彩标签文本的 words_set 取交集，然后对每个文本 words_set 减去该交集，得到属于某色彩标签的不同于其他标签的一组字词集合，称为剩余集合。最后，对各个色彩标签的剩余集合，只保留集合中所有词拆分得到的单个字，再查字典得到其 token，最后需

要加 padding，对齐堆叠成张量之后输出。这样处理的原因后文将详细介绍。

c） 注：因为 batch_size=1，单组数据作为一批数据，所以全部数据的输出形状是多一个维度的。所以还需要一个函数处理。可见下文。

③ taobao_preprocess_func 函数，输入 TaobaoDataset 输出的一组数据，去除多余的维度，并转移到 cuda 上加快计算速度。

经以上流程处理，一批数据（即一组图片-标签集）的输出格式汇总如下：

1. img_stack ((num,3,224,224) tensor) 图像张量
2. labels_ts ((num,) tensor) 图像标签，用序号表示，序号与 optional_tags 同序
3. optional_tags ([“红色”, “白色”] list) 原始标签，如左方例子所示
4. phrase_ts_padded ((class, longest len) tensor) 各个标签处理后的 token 序列
5. phrase_len_list ((class,) tensor) padding 前 token 序列的长度
6. (if for test) img_names 图像文件的文件名 用于生成 test 结果
7. (if for test) directory_name 数据文件夹名 用于生成 test 结果

模型训练

① 设置日志记录器便于实验后查阅结果

② 设置模型结构与训练相关的各个参数。模型在 models.py 定义。

③ 实例化：训练/验证数据集及加载器、模型、损失函数类、优化器等。

注：损失函数类和判断正确率的类分别为：**criteria.py** 中的 **Taobao_CrEntr_Contrast_With_Logit** 类和 **Correct_vs_Total** 类

④ 涉及张量运算的数据预先加载进 GPU 中。

⑤ 读取一组数据，进行前向传播，计算损失函数，进行后向传播。

⑥ 重复上述过程迭代训练

模型评估

① 在验证集上进行无梯度记录的预测，并估计模型的性能，包括 AC（图片标签预测准确率）与 EM（某一版型服饰的全色彩款式图片标签均预测正确的概率）。

② 判断是否应当终止训练，此处设定为一旦 AC 下降则终止。

③ 每轮随时保存模型权重。从 early stopping 原理来看，一般使用倒数第二次迭代得到的模型会取得更好的效果，所以应当随时保存权重。

2.4 测试集上测试并输出 json 格式结果 TEST_SET_PREDICT.py

构建一个字典用于存储测试结果。模型的具体的测试算法与模型验证中的算法有不同，后文将具体介绍。

三、 算法原理与实现过程

3.1 数据处理中的算法原理与细节介绍

1. 对图像：

使用 medium 数据集进行训练和测试。对读入的 PIL.Image 格式照片进行如下的处理：由于图像长宽各不一致，所以先 Resize 到长边为 225，这可由 max_size 来设置，再中心裁剪到（224,224）的尺寸，不足的尺寸会 padding 为 0。再转化为张量，最后用 ImageNet 上均值和方差进行标准化。如下代码所示。

```
transforms.Resize(square_size[0], max_size=225),
transforms.CenterCrop(square_size),    # can do padding if too small
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

这样方便输入经典的预训练卷积神经网络并且不会过分失真。

2. 对文本：

如前文流程描述，对每一组文本标签拆分获得字、词及所有其可能的组合，遍历 n 个标签生成的该集合取交集，最终对每个集合减去该交集，得到的剩余集合可以视为去除了多数与色彩无关的词语的相对纯净的一个表征色彩特征的词语集合，最后对该集合中每个词都拆解成字构成新的字集合，查找字的 token，加 padding 排列成 phrase_ts_padded，即为预处理完的文本数据。

此处，除了要去除共有的词语（视为无关词语）以外，还拆分成了单个字，抛弃了词语的形式。这是因为，笔者在观察数据实例的时候发现，色彩词基本上每个字都可以代表一种颜色，不需要组成词语，譬如“海军蓝”“卡其色”“橄榄绿”中，如果出现了“海”、“蓝”甚至“天”基本就模糊地映射到了蓝色的色彩，而“其”“卡”基本就映射到了卡其色的色彩，“榄”“橄”“绿”则基本就映射到了绿色的色彩，极少有例外的情况。当然，这只是一个

大致的直觉判断，具体还是因为后续实验中这么做比词语搭配单字甚至只有词语的预处理方法效果好得多。所以根据先验的观察，我们可以取巧地只使用单字来进行预测。另一方面，按照笔者的理解，与其把所有可能的词语都建立臃肿冗余的 Embedding 映射（且 nn.embedding 的学习能力也比较有限），还不如尽量缩小有效的 token 规模，只取单个字。举例来说，经过统计，训练样本中只出现过几次“黝黑”的标签，那么对它建立 token 来训练唯一的 embedding，极少的数据量，可能很难学到比较理想的表征，但是拆开“黝”“黑”，就更容易推理，数据量相对也会更多（“黑”出现了几千次），更何况“黝黑”和“黑”基本没有色彩上的差别。加上此处笔者的训练过程是同时训练文本-图像智能体的，同时训练会大大增加收敛的困难程度与模型崩溃的可能性，笔者希望文本特征提取的模型相对简单易收敛一些，这样训练会更快甚至更好地达到预期的效果。因此不应当让 nn.Embedding 层的输入 token 太过复杂。综合本段的几点想法，笔者选择用拆解出的单字来进行预测。

3.2 模型算法原理与细节

1. 图像色彩特征提取模型：

① ViT-b-16, 预训练模型, 输入（B, 3, 224, 224）张量图像，输出 1000 维向量。

使用 torchvision 0.12.0 新加入的 vit 模型库来加载模型（由于许多算力资源平台都没有安装新版 torchvision 0.12.0，所以笔者在本代码文件夹中也放入了 torchvision12 库文件，这样就可以避免版本带来的问题）

② nn.ReLU()

③ nn.Linear(1000, 200, bias=True) 输入 1000 维向量，输出 200 维作为特征向量。

原理：这样设计的好处是利用了 ImageNet 上预训练的 ViT 模型的初始化参数，其引入了自注意力机制，是近年来计算机视觉领域性能最好的模型之一，笔者认为该模型已经能够对商品图片有一定的语义理解，所以只需要微调就可以运用到本任务。另外，经过测试，200 维作为文本与图像的公共特征维度能够取得最优的性能。

2. 文本色彩特征提取模型：

① 单层 nn.Embedding(vocab_size=5739, embedding=200, padding_idx=5738)

输入 dataset 给出的 phrase_ts_padded 及长度标记，能够查询得到每个标签中预处理分离出的单字的 embedding。形状: [标签数，单字数，200 维特征]

② 对形状为 [标签数，单字数，200 维特征]的张量的第二维度进行求 mean 降维，得到[标签数，200 维特征]形状的张量，即可作为每一个标签的色彩特征。具体是对每一个标签中取其长度 len，对前 len 个输出的 200 维特征进行求 mean，得到该标签的 200 维特征。最后对所有标签的特征堆叠生成张量即可。

原理：单独一层 Embedding 给出的 200 维向量作为文本图像公共的色彩特征，已经能够达到 AC>92%，EM>84%的效果，可见简单的模型也能够达到尚可的性能。笔者的理解是，用预处理后的单字输入，单字已经能够表征一定的色彩，而如果某些单字譬如“色”反复出现或者“黝”很少出现，与色彩的预测关系不大，就会被调整到趋近于零，因此在取单字的 mean 时基本不会产生作用，而“黑”经过大量的训练，已经调整到比较理想的表征状态，所以能够很大程度影响 embedding 向量在特征空间中的方向。总的来看，这样的单层 Embedding 效果较为理想。

另外，初始化时设置的词汇规模 5739 是偏大的，但是在实际的训练中，由于单字的规模比较小，所以真正被激活并优化的 nn.Embedding 的列表只有一小部分，因此设置影响不大。

3. 前向传播：

文本和图像输出的特征向量会分别经过单位化处理，只保留方向特征，即除以二范数，为避免二范数过小，分母为 0，除数加上 1e-5。

特征向量分别记作：img_emb (m, 200), text_emb (n, 200), 输出相似度矩阵：

$$similarity = img_{emb} \cdot text_{emb}^{T}$$

形状为（m，n）。

4. 损失函数：

总的 Loss 函数由三部分得到：

可表示为：$Loss =$

$$w * Loss_{contra}(img_{emb}, label_{img}) + w * Loss_{contra}(text_{emb}, label_{text})$$

$$+1 * Loss_{CrossEntropy}(similarity, label_{img})$$

①$Loss_{contra}$（GroupColorContrastFunc）运算规则如下：

$$Likelihood = emb \cdot emb^{T}$$

$$LabelMat = \frac{\text{int}(label - label^T == 0) - 0.5}{0.5} - I$$

I 表示对角阵，int 表示逻辑矩阵数值化（True=1，False=0）。在$LabelMat$上，行列所代表的两个数据若为同类则为 1， 否则为-1，自身对自身为 0。

返回：

$$Loss_{contra}(emb, label) = -mean(multi(\ LabelMat, Likelihood))$$

其中，multi 为逐元素相乘, mean 为取均值得到一个标量。

$Likelihood$的行列表征对应项与项之间特征向量的相似度，逐元素相乘 multi($LabelMat, Likelihood$)两矩阵再取相反数，则表示：对同类特征之间的相似性加大鼓励，而对非同类特征之间的相似性增大惩罚，可使其疏远。可使同类特征之间的相似性接近。

对 text 与 image 的 embedding 都分别计算如上的 Loss，text 的 label 即为 tensor(range(n))。目标是使文本与图像的异类表征分离度加大，同类表征分离度减小。

②$Loss_{CrossEntropy}$（Taobao_Group_Cross_Entropy_With_Logit）

输入 similarity 和 label。计算前向得到的 similarity 给出的分类预测结果，再计算交叉熵。

Similarity 每一行为每一图片对所有标签分别的相似度，计算预测结果为：取 log(softmax(Similarity))后取对应行的对应 label 所指向的列，即为真实类别的对数概率，取负数求和即为交叉熵。这其实是交叉熵的简便表示方式，因为y_{gt}是 one-hot 的。

该损失函数用于将 text 与 image 的特征空间拉近，鼓励其对相同的色彩表征出相近的向量。

③ w=0.8 参数用于调整三个 Loss 之间的比例关系，增加调参的灵活性。

综上所述，笔者设计了一个三个成分的损失函数，各司其职，一个处理文本，第二个处理图像，第三个拉近图像与文本之间的距离。三者联合引导模型更好的获得对服饰色彩的识别能力。

5. 预测算法：

预测算法是笔者设计中最能提升 EM 指标的一部分，笔者设计了聚类算法，

将预测 EM 值从 78%左右提升到了接近 85%，而 AC 也会在高位小幅增加。

本实验设计了三种预测算法，

① 简单说明一下 predict 和 predict4vote，它们 EM 效果并不好，但是用于训练验证能够有较高的运算速度。

Predict4vote 是直接对前向传播得到的 Similarity 做每行的 softmax 之后求每一行的 argmax 得到预测结果。

Predict 则是利用了本任务的一个限制条件：每一组的每一个标签都一定有多于 0 个的对应图像。于是我先对 softmax 的结果求每一个标签下，属于该标签的概率最大的商品，即求每列的最大项。具体算法为：先求第一列 argmax 得到一个行的下标，即 img，删除该行，同时维护一个包含所有 img 下标的列表，一并删除该 img 下标，保存该 img 值与对应 label 值，再求下一列的 argmax，再删除…如此处理得到所有标签的最可能的商品图像，最后再对余下的商品求各行的 argmax。得到的结果输出即可。

② predict_cluster 用于测试集预测，不用于训练，很耗时。

先对 img_embedding 进行 n（n 为 tag 数目）类的模糊聚类。

初始化：用 similarity 的软最大概率找到每一个标签最可能的图像，对应图像的特征向量作为初始聚类中心。

迭代如下过程：

a)隶属度：dist 为欧氏距离（+1e-5），mj 为第 j 类的聚类中心，xi 为图像特征。

$$miu_j(x_i) = \frac{\frac{1}{dist(x_i, m_j)}}{\sum_l \frac{1}{dist(x_i, m_l)}}$$

b)聚类中心：

$$m_j(x_i) = \frac{\sum_i miu_j(x_i)^2 x_i}{\sum_i miu_j(x_i)^2}$$

直到聚类中心变化小于特定值或达到最大迭代次数。

下一步进行标签匹配。暂且将未匹配的聚类标签称为聚类标签。

笔者将收敛的聚类中心与文本的特征向量进行内积。内积获得相似度矩阵的公式与前文类似。得到的相似度矩阵通过对每一个真实 tag 求最相似的聚类标签来使得两个系列一一对应。在算法中，由于前面选完的 tag 不能再选，所以也要随时进行删除 row 和 index 的操作，具体的算法与 predict 中描述的算法类似。

这样做基于的假设首先当然是每一个标签都对应至少一个图像，其次也假设了训练好的模型，能够较好分开不同色彩的图像，加上图像特征与文本特征是对齐的，因此聚类中心是能够反映一类图像的某种统计特征的，应当也能够与文本特征进行对齐。这就是聚类与标签匹配算法进行预测的大致假设，后续实验表明这一假设是没有问题的。相比使用前两个比较简单的预测算法，该算法能够大幅提高 EM，即能够以更高概率全部分对某一组的标签。

四、 实验结果与分析

笔者前期做了许多其他模型结构的实验。

4.1 如下所示为 ResNet18+RNN 的训练结果随轮数与参数设置变化的记录：

```
word_embedding 100 common_embedding 10  lr 3e-5  ac\em = 89\68 最次优
em 50  fea 20  lr 1e-5  acem = 77 46 | 83 56 | 85 59 | X
20 5 3e-5  77 47 X
200 10 3e-5 | 8824 6644 | 8992 7026 | 90.36 71.20 | 8944 6756 次优
200 100 3e-5 | 8922 6863 | 8950 6882 | 9023 7051 | 8999 7057 优
200 200 3e-5 | 8921 6794 | 9009 7101 | 9000 6994 次次优
300 100 3e-5 | 8936 6869 | 9033 7076 | 9013 7007 次优

resnet50 似乎效果会更好些，需要强的显卡
200 100 | 8894 6882 | 9066 7301@ |

transformer

双向RNN
LSTM?*2*2

集成学习
```

训练时采用 predict4vote 预测，因此 EM 不高。

总体可以看到 resnet18+RNN 的效果 AC 勉强能达到 90%。已经有上限。

4.2 后来采用 MacBert base+ViT-b-16 进行实验，发现效果更差，AC 只有 78%，说明模型没有学到有用的表征。可能是因为 Bert 只适合处理长文本前后的前后文关系与一词多义等的问题，而不适合处理本任务这样的词语分类的问题。因此考虑还是去学习一个 embedding 或许会更好。也可能是因为模型同时进行训练调整，两者很难同步，甚至是模型崩溃。另外，也有可能是模型过拟

合了，因为模型在训练集上达到了大于 90%的正确率，但在测试集上降低许多。

4.3 于是笔者痛定思痛，将之前写的对文本进行预测的所有复杂的模型结构（LSTM、RNN、GRU、BERT、MacBERT）都删了，只保留了 embedding 层这一层用来学习文本特征。此前，笔者的数据预处理是将文本中的词语与单字乱序输出的，笔者经过深思熟虑，删除了输出的词语，只保留单字。另外，笔者尝试改变了图像预处理 transform 的流程，原来是直接形变 resize 成（224,224），存在失真，之后改为了：先等比例放缩再加 padding，减小形变。图像的模型仍选用 ViT，因为图像相对更复杂难以简单化处理。

经过以上的调整，结果模型的性能在验证集上达到了 AC94% EM80%，

经过聚类测试，在网站上达到了 AC>92% EM>84%的效果。是笔者所训模型中最好的一个结果。

五、 代码引用声明

1. 为避免部分平台无法提供最新版本的 torchvision 0.12.0，代码包中直接保存了 torchvision12 的库文件包以方便调用，该代码来自 www.pytorch.org，特此说明。

2. util_func.py 中的日志记录函数 Logger 引用了一位博主，已经在函数声明前标注出来源。

3. 大部分代码文件在网络学堂的代码模板基础上进行全面修改得到，重合率应当不高。但是保险起见，代码中的许多函数或类声明前都标注了“可能参考了网络学堂的模板代码”的注解。

六、 最优模型权重 下载源

保存在云盘上，网址：https://cloud.tsinghua.edu.cn/f/1157c4431ee04cc79002/