智能计算实践第三次实践——图像变化描述

姓名：文浩宇

学号：1120213407

班级：07022101

教师：鉴萍

2024 年 5 月 19 日

智能计算实践第三次实践——图像变化描述

1 实践目、要求与简要说明

修改 Baseline 使之能接受多幅图片的输入，并且实现对变化的描述。说明：提交的部分不够完整（源码、*pdf* 报告已提交，已经满足提交、实践的要求了）。对于尚未完成部分将会继续在*https://github.com/Haoyu-Wen/Homework3*部分更新（更新时间可能不定，但是提交部分尚未跑完的结果一定会近期上传完毕），如果老师对尚未完成部分感兴趣，可以前往该仓库查看。评分可以仅按照提交内容评定，否则对于已经按时提交的同学确实不公平；但是如果老师对后续结果满意还望老师适当（折扣）加分。提交的内容包括低阶、高阶模型源码以及 *checkpoints*。尚未完成的部分——将 *change* 改为 *1* 后得到的数据，重新跑一遍（尚未跑完，提交的是 *change* 为 *3* 的完整结果）；使用 *levir-cc* 数据集重新训练；使用 *resnet101* 做特征提取（提交的是使用的 *resnet50*）。如果后文提到的复现、缺失必要内容可以去仓库找或者要求我重新提交。

2 实验过程

2.1 代码修改

本小节简述修改思路。对于实验要求的接受多幅图像的输入，在本次实践中暂定为 3 幅图像的输入。高阶和低阶都进行简要介绍——不过都表现不佳。之所以两个都做是一方面是因为模型表现不佳，另一方面是所谓的高阶，也没改什么网络结构，不知道能不能称得上是“高阶”，但是确实是满足了训练使用两相图像，推理接受多幅图像的要求，再者，修改高阶和低阶的代码量不是特别大（其余杂项倒是比较麻烦——数据收集、处理）

特征提取 遵循 baseline 的做法，使用预训练的 CNN——ResNet（不过用的是 resnet50 而非原文的 resnet101——因为跑了一些实验后才发觉自己没有使用原文的 resnet），去掉了后面的线性分类器。不过，由于一些因素，特征提取放在了 MCCFormers 类里面，没有预先提取，导致训练、推理的时候需要进行特征提取（速度减慢且需要更多显存了...）。

低阶 首先低阶部分，需要修改模型能够接受多幅图像输入，训练时使用多幅图像输入。修改思路较为简单，主要是修改 encoder 部分，只需要将代码中涉及到 img1 和 img2 的地方添加一个 img3 重复一下即可，但是对于 decoder 部分，需要修改 feature_dim_de 以适应 encoder 部分产生的输出作为输入。

对于 MCCFormers-S，baseline 是将 img1 和 img2 拼接起来输入 transformer，我们可以仿照思路，将 img3 同 img1 和 img2 拼接起来一起作为输入即可。

而对于 MCCFormers-D，baseline 采用的方法是使用某幅图像去 query 另外一幅图像。此时我们需要处理三幅图像，于是有两个思路，一个是拿每幅图像去 query 另外两幅图像（例如 img1 query [img2,img3]）, 另一个思路则是拿相邻的图像互相 query，例如 img1 query img2，img2 query img1，img2 query img3，img3 query img2，直观上的物理意义就是相邻两幅图片的差异而第一种思路则物理意义没那么明显。

在代码中对应的模型为 MyMCCFormers-S/D-1/D-2。

高阶 高阶的要求是模型能够接受多幅图像的输入能描述多相图像变化，但是训练时仅使用两相图像。思路如下：训练时仅使用两相图像，也即处理多相图片时网络结构参数共享才能达到要求（最直观的想法），于是当 img3 输入时，对于 MCCFormers-S，将 img1 和 img2 按原步骤处理一遍，将 img2 和 img3 按照相同步骤再处置一次，得到两个 memory。而对于 MCCFormers-D 也类似将 img1 和 img2 按原步骤处理，img2 和 img3 也采取相同的处理，同样得到两个 memory。在训练中由于只有 img1 和 img2，则只有一个 memory，利用这个 memory 去 decode，更新参数。在 eval、test 或者是推理的时候，则由于多了 img3，返回了两个 memory，分别是 img1 和 img2 之间进行 qkv 运算的结果和 img2 以及 img3 之间 qkv 运算的结果，这时候利用这两个 memory 去解码生成数据。我尝试了将两个 memory 相加再取平均以及拿两个 memory decode 两次的做法，从直观意义上来说后者才是比较符合物理意义。但是可能由于数据量较小的情况，两者表现都不佳。

高阶的做法，就是单纯的 encode 了两次，虽然满足了要求——训练时使用两相图像，推理时能接受多幅（以 3 为例）图像，但是其实没什么特别的，所以感觉自己所做的“高阶”也算不上“高阶”。因为这和在推理的时候，进行两次 memory=encoder(img1,img2),l=decoder(memory) 区别不大，区别就在于两个 memory 是通过网络一次性产生的，而另外一种是在循环遍历的时候做的用网络推理了两次；亦或者说就单纯是网络能否接受多幅图像的区别。当然，这只是我的高阶的做法，可能过于简单，显得不像高阶的。或许可以在网络结构上再搞一些花样...

需要说明的是，高阶的模型在代码中命名为 AdvanceMCCFormers-S/D，如果 img3=None，也即采用 2 相图像（训练）返回 memory，而如果 img3 不为 None，也即采用多相/3 相图像（低阶训练，高阶/低阶推理），返回 memory1，memory2。在代码中 AdvanceMCCFormers-S/D 作为低阶训练与高阶训练都可行（取决与 img3 是否为 None）。Advance 的名字有所误导故特殊说明（也可作为低阶）。

2.2 数据集

本次实验的数据通过 Baseline 提供的数据生成器生成。生成三幅图像，每相邻两幅图像之间有 3 个变化（变化遵循 baseline 的设定）包括以下类别添加、删除、替换、移动。利用 baseline 提供的方法，产生 6 个变化，取原图作为第一幅图，取产生三个变化时的图片作为第二幅，取产生所有变化后的图片作为第三幅。共产生 <img1,img2,img3> 5253 条，其中 90% 作为训练集，10% 作为测试验证集 (也即 4727+526)。

2.3 结果与改进措施

模型表现较为差劲，模型似乎并没有观察到图片差异——回答的内容与变化无关，此外模型对于许多不同测试的图片组采用相同的回答。个人猜想很可能是数据太少了，导致模型训练不够，baseline 产生的数据应该远大于我产生的数据集——baseline 里面 generate_images.sh 里面循环是 0 到 60000 即使中途许多图片可能不合格，但是也远大于我的 4726+526 *train+test*（产生数据太耗费计算资源、时间了）。具体结果见*./code/eval_results* 文件夹。说明：下列各组改进措施得到的模型表现仍然不好，可能有略微改善。

数据增强 由于数据较少，采用数据增强的方式来获得更多数据。具体做法为，将原来的 <img1,img2,img3, caption,caplen> 数据元组逆序，再添加到数据集里面去，也即添加了数据元组 <img3,img2,img1,

```
reference:
There is no longer a small gray rubber sphere .
 The large red rubber cube has disappeared .
 A new large green metal cylinder is visible .
 The large cyan metal sphere has been moved .
 The small blue metal sphere is in a different location .
 Someone removed the large brown rubber cube .
############
answer:
The large purple rubber sphere is missing .
 The large purple rubber sphere is missing .
 The small purple rubber cylinder is missing . A new large purple metal s
 The large purple metal sphere was moved from its original location .
 The large purple metal sphere was moved from its original original locat
 A new large purple metal cylinder is visible . there . A different locat
 A large purple metal cylinder .
&&&&&&&&&&
```

图 1: 产生的结果（AdvanceMCCFormers-S，decode_twice,2 相图片训练），详情可见*./code/display.ipynb* caption*,caplen*> 到原数据里面去。量化指标使用的是 blue-score，由于模型仍然表现不佳，分数都一样（采用了 smooth-funtion，都为 0.178），不过人工简单看了一眼回答的 json 文件的话，会发现可能稍微有一点提升——表现为重复的句子没那么集中于某几句了，但是仍然回答不好。

encode_twice　之前处理低阶的时候，是将三幅图的变化（相邻两幅图 3 个 change，也即总共 6 个 change）一起输入给 decoder 训练。考虑到描述其实是两部分，一部分是第 1 幅和第 2 幅的变化，另一部分是第 2 幅和第 3 幅的区别，直接拼接起来就没有利用到相邻两幅对比的意义了，不过直接拼起来对于低阶模型（即 MyMCCFormers-S/D, 只返回一个 memory）比较合理。对于使用多相图片训练的低阶 AdvanceMCCFormers-S/D，会返回 memory1 和 memory2，将描述拆成两部分就比较合理了），分别

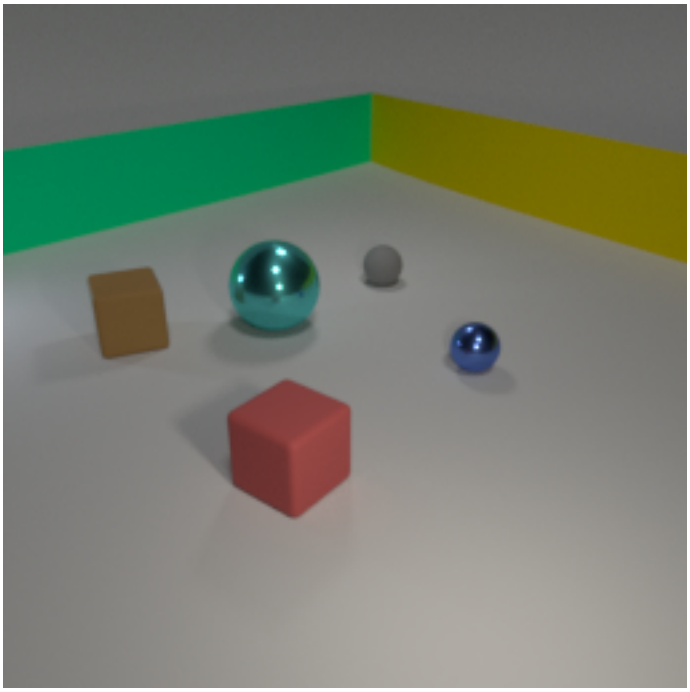

图 2: before

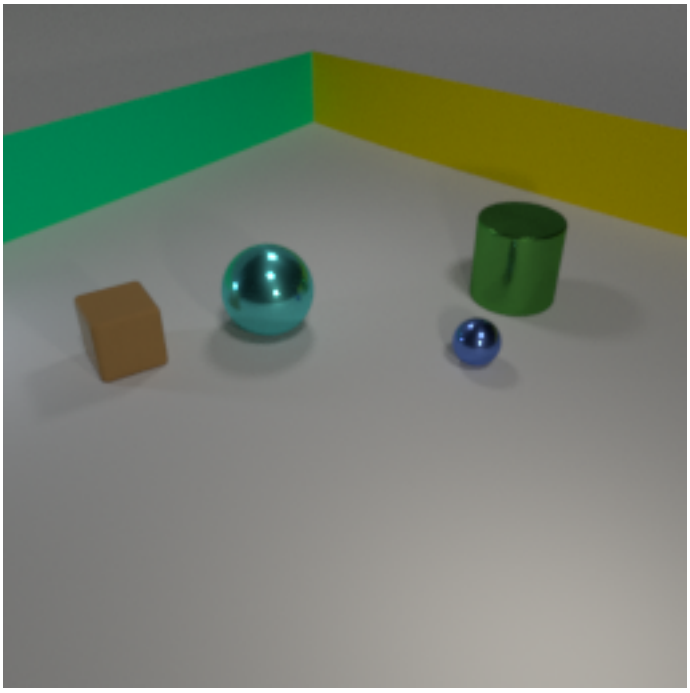

图 3: during

将 memory1 和 memory2 输入 decoder 然后更新（有点类似“高阶”的做法了，只不过一个循环更新两次。不过其实效果也看不出明显变化——数据量可能实在是不够？

decode_twice　在推理的时候，上文提到 AdvancMCCFormers-S/D 在接受 <img1,img2,img3> 输入时会返回两个 memory。进行了两种处理，一种是将 memory 直接求平均，另一种是将两个 memory 分别 decode。

beam_search　在评估时，解码的时候采用了 beam_search，以减少回复重复、同质化的情况，让模型不那么依赖 <start> token。（其实代码里面还有 top-k 解码，但是效果都很差，于是最终决定使用 beam_search）

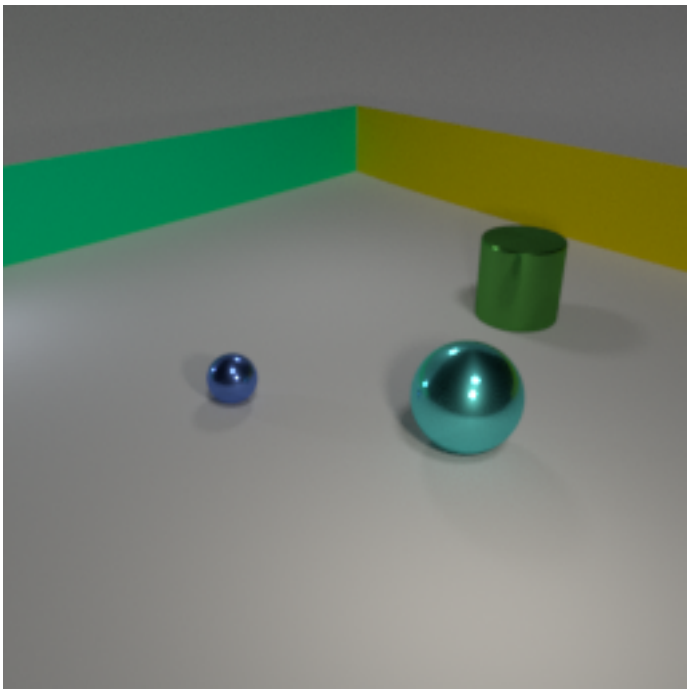

图 4: after

更改数据集 原有的数据集可能确实数据不够，并且处理的是 blender 渲染的图像。而 levir-cc 数据集是遥感图像。之后准备高阶利用 levir-cc 数据集重新训练一个，但尚未完成。后续训练完成后将更新 GitHub 仓库（代码、报告、checkpoint）。

降低任务难度——change 数改为 1 在2.2提到，两幅图之间的变化为 3。修改两幅图之间的 change 数为 1，理论上来说降低了任务的难度。这里称不上是“改进措施”，这里设置这个栏目的目的是为了了解降低任务难度后，模型的表现效果会不会更好，也即模型本身能力是否能处理提出的任务。（尚未跑完）

其他 其他还有不少可能会产生提升的方法，数据方面：将数据重复训练（同样类似数据增强）；将图片旋转之类的数据增强的方法；可能采用更多不同类型的数据–增强模型的泛化能力，防止模型回答重复的句子，过度依赖 <start>，不过可能涉及模型不好收敛的问题–或许更多数据也能解决。模型架构方面：decoder 部分或许也可以采用预训练的模型，只不过限于算力，无法尝试；encoder 选用不同的 CNN，同样限于算力，也无法实施了（做本次实践的时间比较晚，跑整个实验得花很多时间）；其余可能的提升就是修改模型了，比如之前 MyMCCFormer-D（低阶的）里面涉及到的哪幅图去 query 哪些图，也算是一个方面，query 后得到的值直接拼接还是进行某些操作后再用某种方式拼接，比如 img1 query img2，img2 query img1，img2 query img3，img3 query img2，将 output12，和 0.5*output21+0.5*output23，output32 拼接起来。改模型架构方面，对于我而言感觉是黑匣子，以上只是我的一些并未实施的思路，如果时间（个人因素，本次实践给定的时间已经足够久了）、算力再宽裕些，或许能做更加完整。此外 eval 的时候缺乏量化指标，目前使用的是 sentence 的 blue score，但是模型表现不好（且对于相同的动作可以有不同的描述/句式），所以 blue score 并不是特别合适——可行的方案除了人工打分，可以使用 GPT 来打分（后续可能会使用 GPT 来评估句子之间的契合度——也可能不会后续更新该部分）。

3 复现说明

数据下载与产生 数据下载链接请前往https://github.com/Haoyu-Wen/Homework3查看。将 dataset 文件夹置于 code 文件夹下，数据存放于 dataset 文件夹。

如果需要自己产生数据，则参照 baseline 的设置，install blender 并配置环境变量。之后则是运行 dataset_generation 下的 generate_images.sh 脚本（注，如果需要复现的话，请不要更改.sh 文件里面的参数部分，但是可以修改循环次数——涉及产生的数据大小）。随后运行 python generate_captions.py。随后运行 python mydataset.py（默认的 changes 是 3，也即两张图片之间的 change 数为 3，复现不用修改）。得到对应的数据集后（默认名字为 CLEVR–低阶的训练数据和 CLEVRhat–advance 的训练数据），利用 mv 指令将这两个数据集移动到 code/dataset 目录下即可。总结如下 (示例):

- cd ./dataset_generation/image_generation
- bash generate_images.sh
- python generate_captions.py
- python mydataset.py
- mv CLEVR ../code/dataset

- mv CLEVThat ../code/dataset

模型训练 模型训练参照 code 文件夹下 train_script.sh 文件夹，每条语句对应不同设定（reverse_data，高阶——advance，encode_twice 等等）。

模型评估 模型评估参照 code 文件夹下 eval_script.sh 文件夹（注意检查 checkpoint 的路径）。此外 display.ipynb 文件可以少量查看 eval 的结果。所有结果见./code/eval_results

某些参数说明 更多详情见代码。**modified_train_trans.py**

- –advance 是否采用高阶训练（传入任何字符串）
- –num_images 训练的图像数（每组），低阶为 3，高阶为 2
- –reverse_data 前文提到的逆序 img 对以扩大数据集的 size
- –encode_twice 前往提到的将三幅图之间的区别描述拆成相邻图片的差异
- –captions_per_image 设为 1（将一组图片的所有 caption 拼接起来了）
- –info 保存 checkpoint 时的信息
- –dataset_name 训练需要的数据集名称

modified_eval_trans.py

- data_folder 数据集位置
- –ckeckpoint checkpoint 的路径
- –model_name 模型名（保存结果需要）
- –dataset_name 数据集名称
- –info 保存 json 结果时的信息
- –beam_size 束搜索大小

文件结构 数据存放于./code/dataset。checkpoints 存放于./code/result 文件夹下。eval 结果存放于./code/eval_results 文件夹下。