深度学习大作业报告

520030910379 刘宜🄵

1. 第一部分：复现开源的深度匹配算法

1.1. 阅读指导文档并配置环境

首先阅读图匹配的相关介绍和指导，并配置相关环境。由于在小作业中，已经配置好了大部分的环境，因此在大作业中，我在此基础上配置了一些额外的包。在配置时遇到了一些版本号不匹配的问题，通过查阅相关文档我也解决了这些问题，同时因为在作业进行的过程中，相关库也在进行更新，某些时刻遇到了一些报错，在升级相关库到最新版本后这些问题也顺利解决。

1.2. 阅读神经网络图匹配算法论文

1.2.1. 深度学习与图匹配的一些背景

首先我阅读了相关的知识，了解到图匹配的求解是一个 NP 难的二次组合优化问题，如何快速精确的求解是图匹配问题要研究的一个重要的内容。

另一个重要的内容是，如何构建相似度，在之前的工作中许多学者提出了不同的数学构造方法，但是普通的数学构造方法无法应对多样化的数据，在 2018CVPR 的一篇文章中，作者首次将深度学习引入到了图匹配问题中，深度学习的方法可以学到更多的内在化特征，解决了这一问题。

1.2.2. 之前的神经网络图匹配方法存在的问题

在 2018 年的一篇文章中，作者使用了这样的结构进行图匹配：

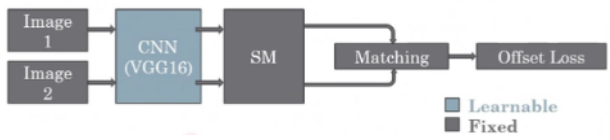

图 1: 早期神经网络图匹配结构

首先对于两张图片，首先通过 CNN 提取图片的信息，接下来通过 SM 求解器来求解图的特征值和特征向量进行图的匹配，最后计算误差，进行梯度回传和参数更新。可以看到网络的设计是非常简单的，但是这对于早期的设计而言已经是非常出色的想法了。

但是这个方法本身存在很多问题，首先 SM 求解器不是一个图匹配的求解器，其求出的解对于图匹配来说仅仅只是一个近似解，前面的部分训练的再好，近似解就决定了模型性能会有损失。同时 SM 求解的成本很高。另一个问题在于损失函数的设计，这里的损失函数是一个 offset loss，其主要是衡量了匹配点在两个图片上的位置，通过衡量两个匹配点之间的 distance 来决定 loss，但实际上，distance 并不能决定 loss，因为对于两张照片，其物体可能有平移有旋转，我们想要衡量的是其是否匹配，因此 distance 并不能说明这一信息。

图 2: *offset loss* 效果举例

文章中也给出了例子说明这一点，比如左图中，马的左耳匹配到了右图中马的右耳，这显然是一个不好的匹配，但是 offset loss 根据其距离较近，给出了很小的 loss，这说明了其缺陷性。

1.2.3. 我对 PCA-GM 算法的理解

PCA 的结构由图给出。

为了解决上面提到的问题，汪学长带领的团队提出了以下的改进方法：

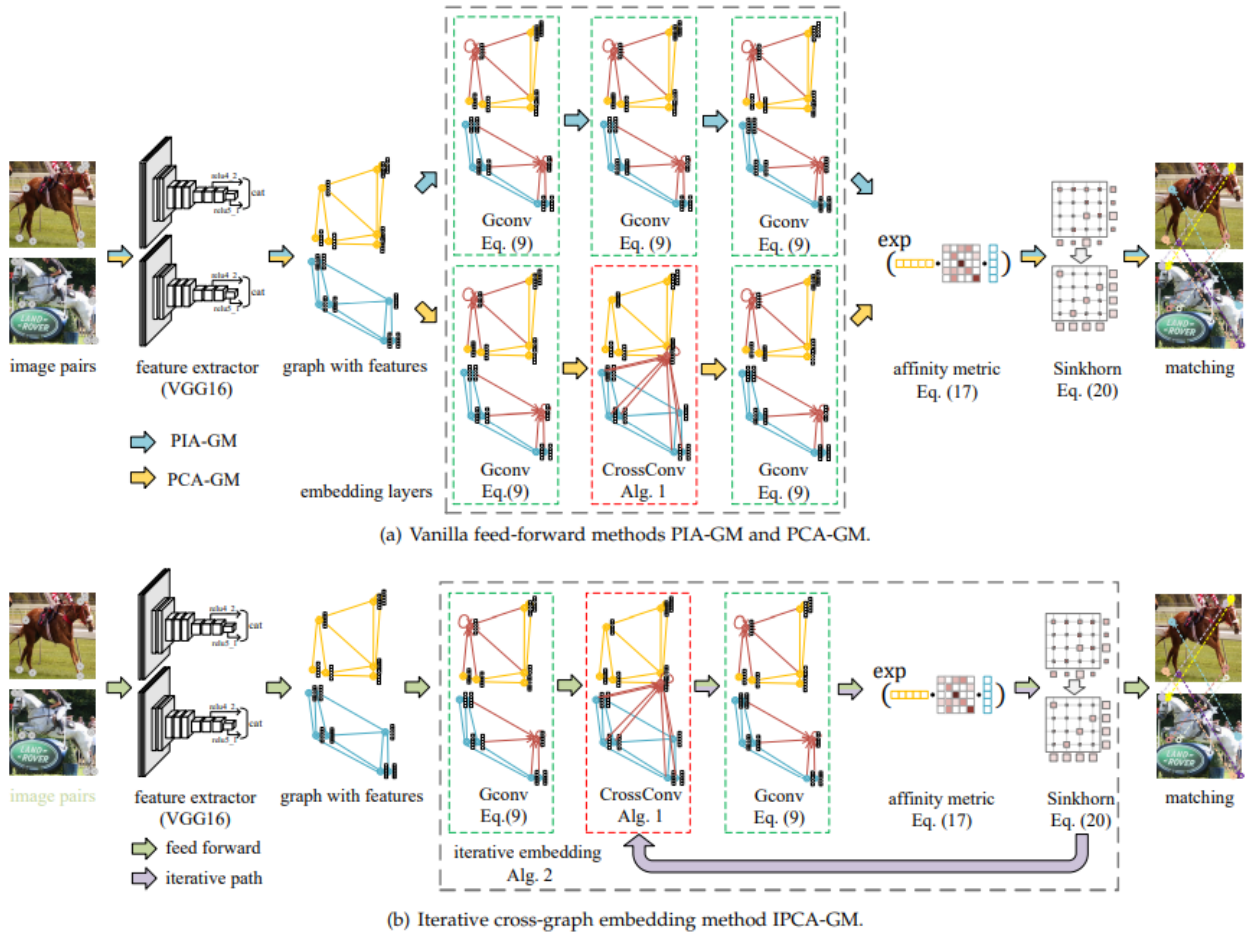

图 3: *PIA,PCA,IPCA* 结构

首先是 SM 求解器的问题，在 PCA 的结构中采取了嵌入网络，这个想法的来源主要是因为要处理图的信息，所以要采用 GNN 进行信息的嵌入，根据嵌入网络的不同形式，作者提出了不同的求解器。

嵌入网络的主要子结构有两种，一种是 Intra-GNN，一种是 Cross-GNN。Intra-GNN 负责图内信息处理，其使用了图内的卷积，将边的信息嵌入到节点的特征向量之中，通过邻接节点之间传递特征信息，就能够在节点的特征向量中嵌入图结构的信息，进而体现图结构的相似度。之后对于节点，我们就可以用相似度矩阵的方法去计算其相似度。

PIA 的方法就是基于 Intra-GNN 设计的，首先对于图片使用 CNN 提取特征，然后使用 Intra-GNN 算 embedding，然后用 embedding 算相似度，之后会得到一个相似度矩阵，那如何得到最终的图匹配结果呢，这里作者使用了 Sinkhorn 算法，对矩阵进行交替的行和列归一化，使其转化为一个双随机矩阵，对这个双随机矩阵我们使用交叉熵计算得到结果，这样做的好处是首先交叉熵是凸函数，其次这样的损失计算方法不会基于像素和距离的概念，这样就避免了以往工作里 loss 函数的问题。这样的提取特征——embedding——得到相似度矩阵——Sinkhorn 算法得到结果的流程，即为 PIA 的图匹配方法。

但是作者在 PIA 的基础上还做了一些改进，因为我们要做的是两个图的匹配，因此作者想到在做 embedding 的时候，采用 Intra-GNN+Cross-GNN+Intra-GNN 的方法，其中 Cross-GNN 的作用是将两个图的信息交叉，通过两个图之间的信息加权传递，使得两个图的嵌入结果尽可能接近，

在之后的匹配中会使得效果提升，这样的结果就是 PCA 的图匹配方法的结构。

在此基础上，我们还可以继续改进，通过迭代的方式，在匹配后再 embedding 加强，再继续匹配，通过这样迭代的方式进行匹配能力的加强，作者通过这样的强化方法优化效果。

1.3. 运行代码复现结果

尝试运行官网给出的样例并得到了结果的复现。

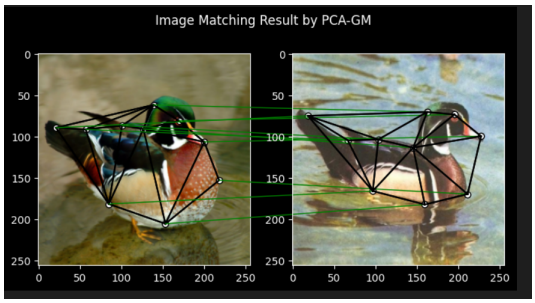

图 4: 结果复现

这一部分的复现还是比较基础，在复现的过程中主要注意修改文件路径等接口即可，在样例里我体会了各个算法的流程，尤其是德洛内三角化的可视化，非常的形象化。

关于损失函数的性能，文章中也给出了相关的数据结果，新的损失函数和结构能够在更大的噪声下保持很好的性能，超越了 CVPR2018 文章中的效果，将 CVPR2018 文章中的损失函数改成新的损失函数，效果也有明显的提升，这证明了新的损失函数设计的正确性。

2. 第二部分：使用国产框架完成深度学习训练、测试

2.1. 准备工作

首先我阅读了作业指导中给出的相关文档和引导，查阅了相关库的官网和文档，了解了数据集以及模型的要求，做了一个总体的初步规划。我选用了 jittor 框架，需要完成数据集加载，模型搭建，巡礼和测试部分代码的编写。

2.2. 数据集加载

数据集加载模块，我主要查看了 pygmtools 中提供的 WillowObjectClass 数据接口。

我创建了数据集类，在初始化函数中创建 benchmark 对象。在 getitem 函数中调用 get_data 函数获取数据，并且参考样例中的流程，对取出的数据进行特征点的计算并且构建对应的边，在得到所有的信息后，将所有的数据一并 return，供其余部分使用。

2.3. 定义模型

模型的定义，主要的工作就是将实例中给出的 pytorch 接口改写为 jittor 接口，这里我主要在 jittor 文档中查阅了相应的接口，对于一些 jittor 未实现的接口，我查阅了相关的 pytorch 接口进行改写。

在做完这一部分后，我发现图匹配实例中其实给出了 jittor 版本的模型实现，我对比了我自己改写的模型和样例给出的模型，基本一致，虽然这里绕了一些弯路，但是自己实现了一个比较正确的模型接口，还是很有收获的。

2.4. 训练和测试模型

模型的训练和测试即为标准的流程，先加载模型，再准备数据集，进行训练，误差反传，测试。在测试时，收集得到结果后，调用 benchmark 中的 eval 函数，得到匹配的精度。

2.5. 遇到的一些问题

在做这一部分时，因为 pygmtools 库更新了几个版本，我遇到了很多次版本不匹配的问题，在饶翔云同学的指导下，我也了解到一些复杂难以读懂的报错其实是版本不匹配的问题，更新版本后即可解决。

在实现时我也遇到了很多次图片的接口不匹配的情况，我选择设置断点，将报错的部分的数据输出查看，最终也是顺利解决了这些代码编写的不规范，构建出了正确的模型。

3. 模型的结果以及优化探究

3.1. Baseline 与预训练

以下是我的基本模型的结果，由于实验室资源很挤，同时中途有人恶意占用资源，所以我有时候申请不到足够的资源，只能进行部分的训练，因此我选择了训练 15 个 epoch 后的结果。

在训练过程中我发现从头的训练，15 个 epoch 太少，模型难以收敛因此效果很差，甚至达不到 50 的 acc，因此我意识到从头的训练在资源不足的情况下不是一个好的方法，我查阅了相关的接口，发现 pygm 中提供了 willowobject 的预训练模型。我采取了预训练，得到了训练 15 个 epoch 时的结果。

这部分中 scheduler 使用了 exp，optimizer 使用了 SGD。我简单调整了一些参数得到了以下的效果。

表 1: *Base* 结果

类别	p	r	f1
car	0.6160	0.6160	0.6160
duck	0.6823	0.6823	0.6823
face	0.8251	0.8251	0.8251
motorbike	0.6513	0.6513	0.6513
winebottle	0.6389	0.6389	0.6389
average acc	0.6827	0.6827	0.6827

加入预训练后也只能达到 0.68 左右的 acc，因此我考虑调整模型来优化结果。

3.2. 优化器更改

首先考虑更改优化器，Base 中使用的是 SGD，查阅后得知，SGD 存在着不小的缺点，首先是其波动性大，下降速度慢，我观察了中间的 loss 数据验证了这一点，同时 SGD 因为其本身的数学特性，不适用于稀疏数据。我猜测这一点影响很大，因为我们得到的相似度矩阵其实是比较稀疏的。因此我更换了一些其他的优化器，并进行参数的调试，最终发现 adam 优化器易于调整且效果较好。

以下是更换了 adam 优化器的结果，取得了 0.9268 的准确率。scheduler 还是使用 exp。

表 2: 更改了 *Adam* 结果

类别	p	r	f1
car	0.8870	0.8870	0.8870
duck	0.8638	0.8638	0.8638
face	0.9996	0.9996	0.9996
motorbike	0.9316	0.9316	0.9316
winebottle	0.9521	0.9521	0.9521
average acc	0.9268	0.9268	0.9268

3.3. 学习率调整策略更改

在调整了优化器后，我尝试更换不同的学习率调整策略，学习率调整主要是在训练的初期，模型没有收敛，此时使用更大的学习率进行学习能让模型快速向收敛靠近，在训练的后期，大的学习率已经不适合模型，因此要调整到较小学习率使模型靠近最优解。

我更换了 ReduceOnPlateau 策略，得到了也很不错的结果。

表 3: *reduce* 策略结果

类别	p	r	f1
car	0.9035	0.9035	0.9035
duck	0.8766	0.8766	0.8766
face	0.9998	0.9998	0.9998
motorbike	0.9122	0.9122	0.9122
winebottle	0.9280	0.9280	0.9280
average acc	0.9240	0.9240	0.9240

3.4. 尝试其他的图匹配算法

在更改了以上的比较基础的参数后，我还尝试了以下 IPCA 算法，因为我直观的想法是，循环迭代后肯定能让信息更加融合，从而使得匹配达到更高的准确率，作为对比实验，scheduler 改成了 exp，optimizer 改成了 adam。

效果如下：

但是实际结果发现效果有所下降，我个人的猜测是相似的信息在多次 cross 后引起了误匹配，或者是因为训练轮次不足一些匹配的信息在融合

表 4: *IPCA* 结果

类别	p	r	f1
car	0.7559	0.7559	0.7559
duck	0.8267	0.8267	0.8267
face	0.9994	0.9994	0.9994
motorbike	0.8794	0.8794	0.8794
winebottle	0.8983	0.8983	0.8983
average acc	0.8719	0.8719	0.8719

后没有来得及分开。我尝试去获取结果的可视化以及 loss 函数的变化趋势，但是针对这个问题我目前也没有想的很透彻。

4. 国产框架的使用体验与思考

这次任务中我使用了 jittor 的国产框架以及严老师团队带领开发的 pygmtools 库，最直观的一个感受就是这些国产的框架文档、参考、示例做的比较全面，但是在论坛、社区等这样的交流平台还没有太丰富的资源。对我而言，当我遇到一些难以解决的奇怪问题时，我都会去相关官方的主页的论坛社区上去看有没有其他人遇到类似的错误，看看网友给出的可行的解决方式，因为错误千奇百怪，官方不可能给出所有的解决防范，用户的讨论和交流是绝佳的补充机会。但是国产框架在这一方面的开发还不够，一方面可能是开发者业务繁重不能精心维护，另一方面可能也是用户不够，问题的质量和数量都很匮乏。

这就让我有了令一种体验，就是国产框架的必要性，因为我们的这些产业相较于国外产业起步较晚，因此很多体系和技术我们都用的是国外的产品，但是随着我们国家的产业迅速发展，在不断挑战最新领域技术的同时，我们也要发展基础的框架、系统、相关的库等产品。在使用过程中，pygmtools 库确实给了我不小的惊喜，无论是指引还是效果还是接口的使用都非常的优秀，这也让我体验到了国产框架的优秀。发展国产基础框架是必要且必须的，在这次的体验后，我也对这些框架和库的底层有了一定的了解和体验，以后如果能遇到相应的机会，我也想要去参与进这样的开发。

5. 参考文献

[1] Zanfir A, Sminchisescu C. Deep Learning of Graph Matching. Computer Vision and Pattern Recognition, 2018.

[2]R. Wang, J. Yan and X. Yang, "Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach," in IEEE Transactions on Pattern Analysis and Machine Intelligence, doi: 10.1109/TPAMI.2020.3005590.