深度学习大作业报告

519030910360 郑文卓

一.简介

图匹配（Graph Matching）是机器学习、计算机科学、运筹优化等学科中的一项基础性问题，图匹配通过构建两个图相似度关系的优化问题，求解图和图之间的匹配关系，进而辅助下游应用。本次大作业我选择了开源的深度图匹配方法 CIE-H[1]，配置 PyTorch 环境复现论文中的训练/测试结果，并利用国产框架 Paddle，将 PyTorch 格式的预训练模型转换成了 Paddle 格式的预训练模型进行验证，另外我还探索了利用 Paddle 的框架复现 CIE-H 的训练过程，以及基于一些新想法对 CIE-H 进行改进探索。

相比任务要求的 baseline，我作出的额外贡献主要有三点：

1. 利用 CIE-H 在 CUB2011 数据集上实现训练功能，而这个在 benchmark 中并没有；
2. 实现了 Paddle 的训练功能，并分析了结果不理想的原因；
3. 探索调节不同区域权重对 Hungarian Attention 机制的影响。

二.配置环境并复现 PyTorch 训练/测试结果

本次实验遇到的第一个难点是配置环境。我并没有采用助教提供的 docker 环境，而是选择自己新建环境并一步步安装。在配置 PyTorch 环境时比较顺利，并成功复现了 GMN，PCA-GM，CIE-H 三个算法使用预训练模型验证的结果（见表 1）。

表 1　在 Willow Object 和 PascalVOC 上测试的平均准确率（%）

model	GMN	GMN B	PCA-GM	PCA-GM B	CIE-H	CIE-H B
Willow	86.94	79.34	90.83	87.44	**94.47**	88.98
PascalVOC	62.49	62.40	64.24	64.78	**69.50**	68.92

(注：GMN B 表示 GMN benchmark，其余同理)

这里的 benchmark 结果摘自 Models — ThinkMatch documentation。

可以看到测试结果基本达到预期目标，这里 CIE-H B 的 Willow 结果较低的原因可能是网站的

benchmark 是 ThinkMatch 训练得到的，而并没有采用原论文的预训练模型。那么，为什么训练得到的结果会与预训练模型测试得到的结果有较大的差距呢？除去硬件上的一些差距外，我在第六部分——对 CIE 改进的探索中找到了一个可能的原因。

随后我配置环境复现了 PyTorch 版本的训练过程，结果如表 2.

表 2　在 Willow Object 和 PascalVOC 上训练的平均准确率（%）

model	CIE-H	CIE-H B
Willow	**89.46**	88.98
PascalVOC	68.79	**68.92**

可以看到结果与 benchmark 非常接近，说明达到预期目标。

训练 Loss 如下图所示：

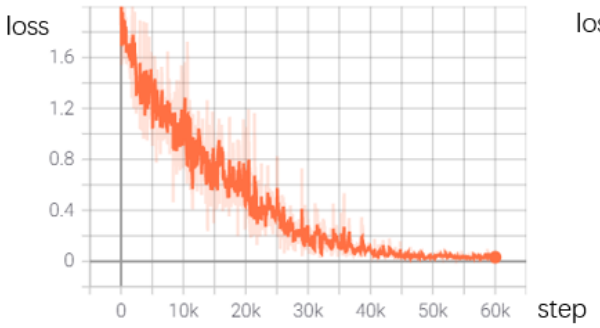

Figure 1 : Training loss of CIE on PascalVOC

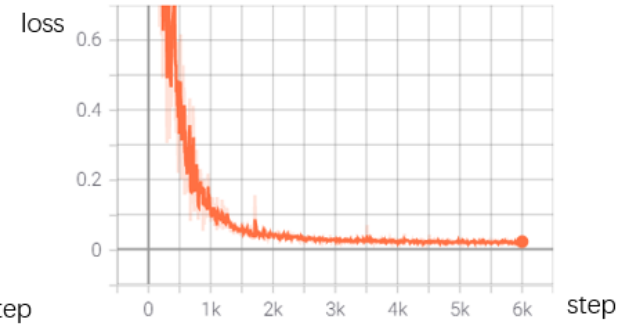

Figure 2 : Training loss of CIE on Willow Object

另外，我注意到 benchmark 中并没有关于 CUB2011 的数据，但是论文中有关于 CUB2011 的实验。于是作为额外部分，我也尝试了在 CUB2011 数据集上实现 CIE-H 算法，结果如下。

表 3　在 CUB 上训练的平均准确率（%）

model	CIE-H	CIE-H B
CUB2011	**94.87**	94.40

这里的 benchmark 结果摘自 [Learning deep graph matching with channel-independent embedding and Hungarian attention | OpenReview](#)。

训练的 Loss 如下图：

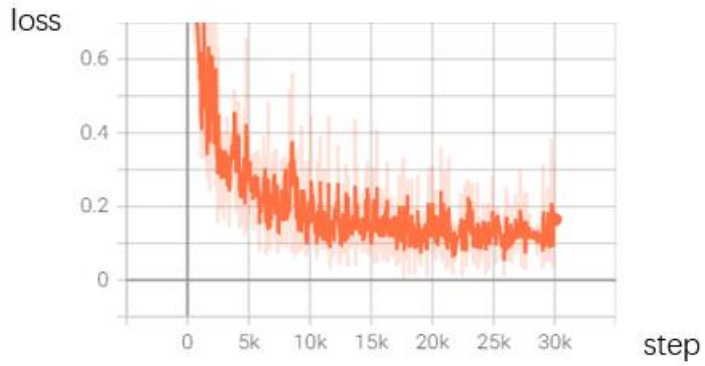

Figure 3 : Training loss of CIE on CUB2011

总体而言，在 PyTorch 上训练和测试的结果都达到了预期，与 benchmark 结果或是论文中结果相近。

三.算法解析

1.论文背景

为了说明 CIE-H 算法的两个核心贡献 CIE 和 Hungarian attention，在此先给出图匹配中的双射问题的定义：给定大小都为n的图$\mathcal{G}_1$和$\mathcal{G}_2$，图匹配需要寻找点的一一对应关系：

$$\max_x x^T K x \qquad s.t. \quad Px = 1 \tag{1}$$

其中$x = vec(X) \in \{0,1\}^{n^2}$是编码两个图之间节点对应关系的 permutation matrix X的逐列向量化形式，而$K \in \mathcal{R}_+^{n^2 \times n^2}$是 affinity matrix。这个问题被称为 Lawler's QAP（Lawler,1963）[2]，并因其 NP-complete 的挑战吸引了广泛关注。近些年提出了许多的算法（Leordeanu & Hebert，2005[3]；Cho et al., 2010[4]; Bernard et al., 2018[5]）大多是基于预先定义的 affinity matrix 的确定性的优化，没有一种学习的范式。直到 2018 年 Zanfir 等人[6]提出了 deep graph matching（DGM），将深度学习与图匹配结合在一起，给图匹配算法提供了一类新的思路。在此基础上 Wang 等人[7]和 Zhang 等人[8]做了许多工作，比如将 QAP（Quadratic Assignment Problem）问题通过 embedding 转化为 LAP（Linear Assignment Problem）问题，使得可以利用 Sinkhorn 算法求解，但是都有一个核心问题：先前的算法都聚焦于节点特征的建模，而边的信息只用于 Graph Convolution Network（GCN）中表征节点连接，并没有很好地提取出边的特征（如长度，朝向等）。为了解决该问题，论文提出了 CIE（Channel-Independent

Embedding）模块，尝试利用边的属性并模拟了 Graph Attention Networks[9]中的多头策略（事实上该策略源自于 Attention is All You Need[10]，核心思路是同时使用多个权值矩阵计算 self-attention 并将结果拼接，将一个节点的注意力分配到其相邻节点集上）。

CIE 的另外一个核心贡献 Hungarian attention 主要是解决 loss 函数的设计问题，在此之前的 loss 函数主要有两种形式：displacement loss[6]和 permutation loss[7]，后者效果较前者好，但论文认为其连续的 Sinkhorn 步骤在测试阶段并不是对 Hungarian sampling 的自然的近似，强迫 Doubly-stochastic Matrix 与最终匹配解完全一致可能会使模型丢失更多的信息。因此论文中提出 Hungarian attention 机制使模型的训练更丝滑。

2.算法框架

算法的框架如图 1 所示，大体流程为：计算出两张图片的 similarity matrix，随后利用 Sinkhorn 算法得到 doubly-stochastic matrix，最后通过 Hungarian attention 机制得到 loss 并反传。

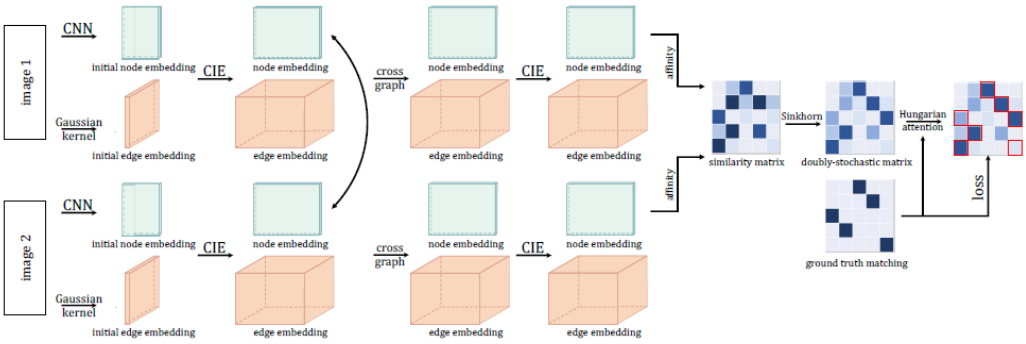

图 1　算法框架

具体而言，对于一张有k个关键点的图片，利用 VGG16 网络提取特征$\mathbf{H} \in \mathcal{R}^{k\times d}$，其中$d$是特征维度，利用 Delaunay 三角剖分算法（将散点集连边为三角网并满足空圆特性和最大化最小角特性，即任意三角形的外接圆范围内不含有其他点，且任意两个相邻三角形构成凸四边形的对角线在相互交换后，六个内角的最小角不会增大）构造 adjacency matrix $\mathbf{A} \in \mathcal{R}^{k\times k}$，另外还初始化了边特征$\mathbf{E} \in \mathcal{R}^{m\times k\times k}$，这里是利用 Gaussian kernel 通过节点特征$\mathbf{H}$得到边特征$\mathbf{E}$。

随后，$\mathbf{H}$、$\mathbf{A}$和$\mathbf{E}$作为输入送进 GNN（Graph Neural Network），其每一层读取上一层的输出，计算下一层的$\mathbf{H}$和$\mathbf{E}$。

$$\mathbf{H}^{(l+1)} = f_i(\mathbf{H}^{(l)}, \mathbf{E}^{(l)}, \mathbf{A}; W_0^l), \quad \mathbf{E}^{(l+1)} = g(\mathbf{H}^{(l)}, \mathbf{E}^{(l)}, \mathbf{A}; W_1^l) \tag{2}$$

其中W_0^l和W_1^l是每一层的可学习参数，f_i和g即 CIE。为了方便得到 doubly-stochastic matrix，论文进一步计算了 cross-graph affinity $\mathrm{M} \in \mathcal{R}^{k\times k}$来评估两张图的点相似性，再通过 Sinkhorn 层得到 doubly-stochastic matrix $\mathrm{S} \in [0,1]^{k\times k}$。

$$\mathbf{M}_{ij} = \exp\left(\tau \mathbf{H}_{(1)i}^{\top} \mathbf{\Lambda} \mathbf{H}_{(2)j}\right), \quad \mathbf{S} = \mathrm{Sinkhorn}(\mathbf{M}) \tag{3}$$

这里Λ是可学习参数矩阵，Sinkhorn 层迭代更新M得到 doubly-stochastic matrix S。

$$\mathbf{M}^{(t+1)} = \mathbf{M}^{(t)} - \frac{1}{n}\mathbf{M}^{(t)}\mathbf{1}\mathbf{1}^{\top} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\top}\mathbf{M}^{(t)} + \frac{1}{n^2}\mathbf{1}\mathbf{1}^{\top}\mathbf{M}^{(t)}\mathbf{1}\mathbf{1}^{\top} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\top} \tag{4}$$

Sinkhorn 层一般被视为 Hungarian 算法的近似，而 Hungarian 算法（匈牙利算法）通常用来解决最优匹配问题，给定一个开销矩阵，核心是找到一种离散化的开销最小的匹配方案。但由于选取最优方案这个操作并不可导，因此利用 Sinkhorn 操作来近似替代，而 Sinkhorn 操作是完全可微的，因此可用于端到端的神经网络训练中。具体而言，Sinkhorn 层对矩阵M交替进行行规一化和列归一化，最后该矩阵收敛为行和与列和都为 1 的 doubly-stochastic matrix S。另外，为了更好地匹配两张图的节点信息，对每张图的节点特征$\mathrm{H}_{(i)}$与另一张图的相似特征进行融合（图 1 中的 cross graph）。

$$\mathbf{H}_{(1)}^{(l)} = f_c\left(\mathrm{cat}(\mathbf{H}_{(1)}^{(l)}, \mathbf{S}\mathbf{H}_{(2)}^{(l)})\right), \quad \mathbf{H}_{(2)}^{(l)} = f_c\left(\mathrm{cat}(\mathbf{H}_{(2)}^{(l)}, \mathbf{S}^{\top}\mathbf{H}_{(2)}^{(l)})\right) \tag{5}$$

3.CIE 模块

CIE 模块的主要目的是将边信息融合到点的 Embedding H中，之前的论文[11]大多是把边特征送进神经网络后和节点特征融合得到信息$m_v^{(l)}$，再与节点 embedding $\mathrm{H}_v^{(t)}$融合得到下一层的结果。

$$\mathbf{m}_v^{(l)} = \sigma\left(\sum_{w\in\mathcal{N}_v} f_t\left(\mathbf{E}_{vw}\right)\mathbf{H}_w^{(l)} + \mathbf{W}^{(l)}\mathbf{H}^{(l)}\right), \quad \mathbf{H}_v^{(t+1)} = u_t\left(\mathbf{H}_v^{(t)}, \mathbf{m}_v^{(l)}\right) \tag{6}$$

这里$\mathrm{E}_{v\omega}$是边(v,ω)的特征，而f_t一般是全连接层，但全连接层会导致高额计算开销，以及梯度反传的不稳定性，因此论文提出了 CIE（Channel-Independent Embedding），具体公式如下：

$$\mathbf{H}_v^{(l+1)} = \sigma\left(\sum_{w\in\mathcal{N}_v} \underbrace{\Gamma_{\mathrm{N}}\left(\mathbf{W}_1^{(l)}\mathbf{E}_{vw}^{(l)} \circ \mathbf{W}_2^{(l)}\mathbf{H}_w^{(l)}\right)}_{\text{channel-wise operator/function}}\right) + \sigma\left(\mathbf{W}_0^{(l)}\mathbf{H}_v^{(l)}\right) \tag{7}$$

$$\mathbf{E}_{vw}^{(l+1)} = \sigma\left(\Gamma_{\mathrm{E}}\left(\mathbf{W}_1^{(l)}\mathbf{E}_{vw}^{(l)} \circ h\left(\mathbf{H}_v^{(l)}, \mathbf{H}_w^{(l)}\right)\right)\right) + \sigma\left(\mathbf{W}_1^{(l)}\mathbf{E}_{vw}^{(l)}\right) \tag{8}$$

这里$\Gamma_{\mathrm{N}}(\cdot\circ\cdot)$、$\Gamma_{\mathrm{E}}(\cdot\circ\cdot)$是一个 channel-wise operator，对每个 channel 独立进行计算，输

出的 channel 数等于输入。我这里对 channel 的理解就是指边（共$k \times k$条边）的特征维度m，每一个维度对应一个 channel（如图 2），共m 个 channels。将对应的边信息/节点信息提取出来并线性转换，随后进行与该维度垂直的操作Γ_{N}和 Γ_{E}。公式后面的$\sigma(\cdot)$代表一个节点传递给自己的信息，保持节点信息一致性。

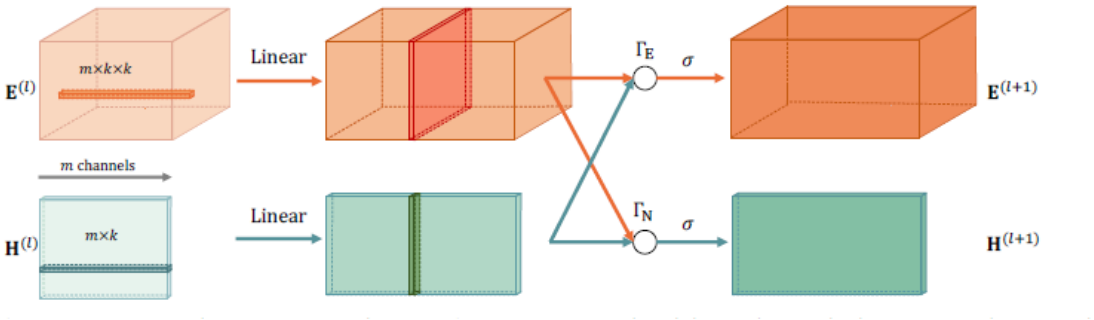

图 2　CIE 模块图示

等式 7 和 8 的具体计算如下，不过在具体实现时需要先添加一些假节点以满足 Wang 等人[7]提出的协议，使得每个 mini-batch 中节点数量达到图像最大范围，这些假节点不会参与训练或测试过程的更新或统计。这里的⊙和⊖代表逐元素点乘和逐对元素作差，$\cdot_{.q}$符号代表取出第 q 个通道的信息。

$$\mathbf{H}_{\cdot q}^{(l+1)}=\sigma\left(\left(\mathbf{A}\odot\left(\mathbf{W}_1^{(l)}\mathbf{E}^{(l)}\right)_{\cdot q}\right)\left(\mathbf{W}_2^{(l)}\mathbf{H}^{(l)}\right)_{\cdot q}\right)+\sigma\left(\left(\mathbf{W}_0^{(l)}\mathbf{H}^{(l)}\right)_{\cdot q}\right) \tag{9}$$

$$\mathbf{E}_{\cdot q}^{(l+1)}=\sigma\left(\left|\left(\mathbf{W}_0^{(l)}\mathbf{H}^{(l)}\right)_{\cdot q}\ominus\left(\mathbf{W}_0^{(l)}\mathbf{H}^{(l)}\right)_{\cdot q}^{\top}\right|\odot\mathbf{E}_{\cdot q}^{(l)}\right)+\sigma\left(\left(\mathbf{W}_1^{(l)}\mathbf{E}^{(l)}\right)_{\cdot q}\right) \tag{10}$$

CIE 的核心优势是什么呢？它基于一个核心假设：节点特征和边特征每个 channel 之间是相互独立的，有两个核心的优势：1.在每个 channel 方向上将节点和边的操作分解（而不是像以前的工作一样把节点和边融合在一起），可以极大地减少参数量；2.受多头策略[9]的启发，CIE 将模型分为d个头，对于每个 channel 中的操作都会重复d次（等式 7 和 8），随后将结果拼接在一起，这样可以让模型关注不同方面的信息，而先前的多头策略将所有 channels 统一对待，构造了一个统一的转换矩阵。

4.Hungarian attention 机制

对于绝大部分图匹配算法而言，其模型的输出都是位于连续域的，即 similarity matrix 的值并没有离散化成 0 或 1 的形式，往往要通过 Hungarian 或 winner-take-all 等算法进行采样得到 0 或 1，其中

Hungarian 算法较为普遍。但是添加了这一步骤就导致一个问题：先前的方法[7]一般会在训练阶段强迫 doubly-stochastic matrix $\mathbf{S}$尽量逼近最终匹配解$\mathbf{S}^{G}$，比如如下的 loss 函数：

$$\mathcal{L}_{\mathrm{CE}}=-\sum_{i\in\mathcal{G}_1,j\in\mathcal{G}_2}\left(\mathbf{S}_{ij}^{\mathrm{G}}\log\mathbf{S}_{ij}+\left(1-\mathbf{S}_{ij}^{\mathrm{G}}\right)\log\left(1-\mathbf{S}_{ij}\right)\right) \tag{11}$$

这样形式的交叉熵损失函数会倾向于让$\mathbf{S}$中的值离散成 0 或 1，并更容易陷入局部最优中。这是因为在迭代次数较少时 loss 会倾向于回传较容易学习的训练样本的梯度，使得某些值接近 0 或 1，而在后面的迭代中 loss 很难更新这些已经接近 0 或 1 的值，导致局部最优。但事实上，这种离散化是没有意义的，因为在测试阶段会添加一步 Hungarian sampling 来对矩阵进行离散化，而 Hungarian sampling 对原矩阵中是否离散并不关心——值是否离散它都一视同仁，所以论文认为上面所述的问题完全可以被避免，因为没有必要在训练时就要求$\mathbf{S}$离散。

论文设想过在训练阶段添加 Hungarian sampling 这一步，但 Hungarian 算法不可微分，很难嵌入进端到端的神经网络训练中，因此论文中提出了 Hungarian attention 这一机制（见图 3）。

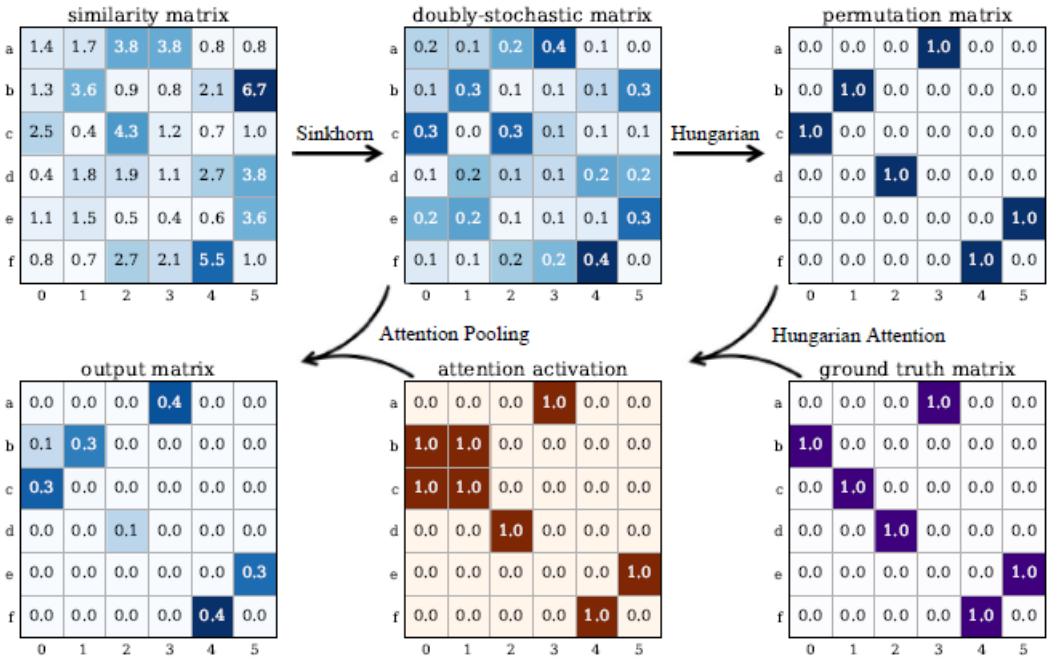

图 3　Hungarian attention 实现流程图

具体而言，对于得到的 similarity matrix $\mathbf{M}$，通过 Sinkhorn 算法得到 doubly-stochastic matrix $\mathbf{S}$，随后利用 Hungarian 算法采样得到预测的匹配解 permutation matrix，将其与 ground truth matrix $\mathbf{S}^{G}$ 取并集得到 sparse link，即图中 attention activation matrix $\mathbf{Z}$。

$$\mathbf{Z}=\mathrm{Atten}\left(\mathrm{Hungarian}(\mathbf{S}),\mathbf{S}^{\mathrm{G}}\right)=\mathcal{P}\cup\mathcal{Q} \tag{12}$$

这里 Atten 即取并集的操作，得到的矩阵Z中为 1 的地方要么是 ground truth 中为 1，要么是预测结果中为 1，相当于一个 mask，让 doubly-stochastic matrix 把注意力集中于这些地方，而不需要考虑其他地方的值，随后经过 Attention Pooling 得到结果 output matrix，最后 loss 函数的形式如下：

$$\mathcal{H}_{\mathrm{CE}}=-\sum_{i\in\mathcal{G}_1,j\in\mathcal{G}_2}\mathbf{Z}_{ij}\left(\mathbf{S}_{ij}^{\mathrm{G}}\log\mathbf{S}_{ij}+\left(1-\mathbf{S}_{ij}^{\mathrm{G}}\right)\log\left(1-\mathbf{S}_{ij}\right)\right) \tag{13}$$

相比之前，Hungarian Attention 机制最大的不同就是关注输出结果局部的小部分，而不关心输出的值大部分是否为 0 或 1。在 Yu 等人[12]的工作中提到离散图匹配问题本质上是对 delta 函数不同排列的拟合，而训练过程就是在连续域近似该函数，但更精确的近似就会导致更高的不平滑度，从而更容易陷入局部最优。因此 Hungarian Attention 关注局部的特点使其缓解了这种现象，毕竟它不需要关注绝大部分的值是否是 0 或者 1。

四.实现 Paddle 版本的测试功能

在配置 Paddle 环境时遇到了一些问题，助教提供的 Paddle 版本的 PCA-GM 算法始终跑不通，进行模型转换的 convert_params.py 也跑不通，最后检查发现原因是 Paddle 的版本不匹配，convert_params.py 中 import 了 Paddle.fluid.* 这个模块，而 Paddle API 文档中提到 fluid 模块会在未来废弃，因此高版本的 Paddle 对 fluid 模块的支持效果不好，需要降低 Paddle 版本，最后将版本从 2.2.0 降至 2.1.0 解决问题。

在修改代码时，核心需要修改的是三个文件：dataloader.py，model.py 和 eval.py。在这其中，重点要实现的几个函数有实现了 CIE 模块的 Siamese_ChannelIndependentConv()，实现了 Hungarian Attention 机制的 PermutationLossHung()，完成数据读取的 get_dataloader()，以及为了支持 CUB2011 数据集修改的 cub2011.py 文件等。

修改过程中，最大的困难有以下几个：

1. 有太多的文件需要修改和关注，往往修改一个文件牵扯到其他数个文件，而且文件路径也要纳入考虑范围，有时候改着改着已经忘记自己前面改了什么。为此我的解决方案是维护两个文档：debug.txt 和 issue.txt，一个是记录自己修改了哪些文件，一个是记录自己遇到的问题以及解决

方案。

2. PyTorch 中的部分函数在 Paddle 中并没有对应的同名函数，比如 Parameter()；有些对应的同名函数用法也不一样，比如 transpose()。这时候 PyTorch 和 Paddle 的官方 API 文档是最有力的工具，查阅文档搜索对应的函数是最好用的办法。
3. 由于代码在服务器运行，且文件非常多，当运行出现问题时 debug 就成为了非常困难的事情，我的解决方案就是 print，print，print！另外，Stack Overflow，Google，CSDN 等网站都是好帮手。

最后实现了 Paddle 版本的测试功能，结果如下：

表 4　在 Willow Object 和 PascalVOC 上测试的平均准确率（%）（Paddle）

model	CIE-H Paddle	CIE-H Torch	CIE-H Benchmark
Willow	93.91	**94.47**	88.98
PascalVOC	68.84	69.50	**68.92**

可以看到 Paddle 上测试结果与 Torch 上测试结果非常近似，说明模型转换是成功的。

五.实现 Paddle 版本的训练功能

作为额外部分，我同样尝试了还原 Paddle 版本的训练功能，主要修改的文件是 train_eval.py。复现结果如下：

表 5　在 Willow Object，PascalVOC 和 CUB2011 上训练的平均准确率（%）（Paddle）

model	CIE-H Paddle	CIE-H Torch	CIE-H Benchmark
Willow	79.47	**89.46**	88.98
PascalVOC	48.99	68.79	**68.92**
CUB2011	75.51	**94.87**	94.40

可以看到结果非常不理想。分析原因时参考了反传梯度部分消失 · Issue

#34419 · PaddlePaddle/Paddle (github.com)：由于 convert_params.py 文件的限制，我目前使用的 Paddle 版本为 2.1.0，但这个版本在运行 Sinkhorn.py 时会出现部分 batch 回传的梯度为 0 的情况，如下图所示：

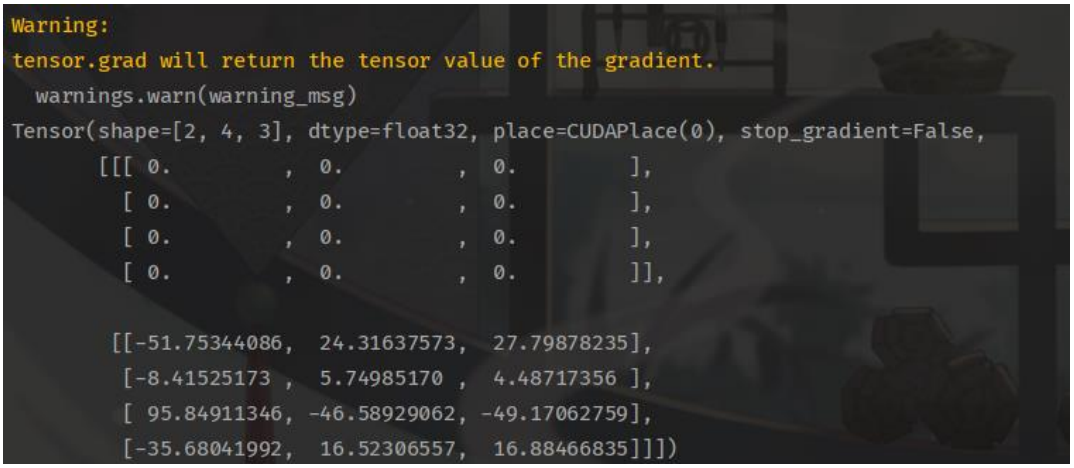

图 3　梯度为 0 图示

于是我尝试将 batch_size 改为 1，但训练时会崩溃，Loss 飙升而 acc 骤降，如下图所示：

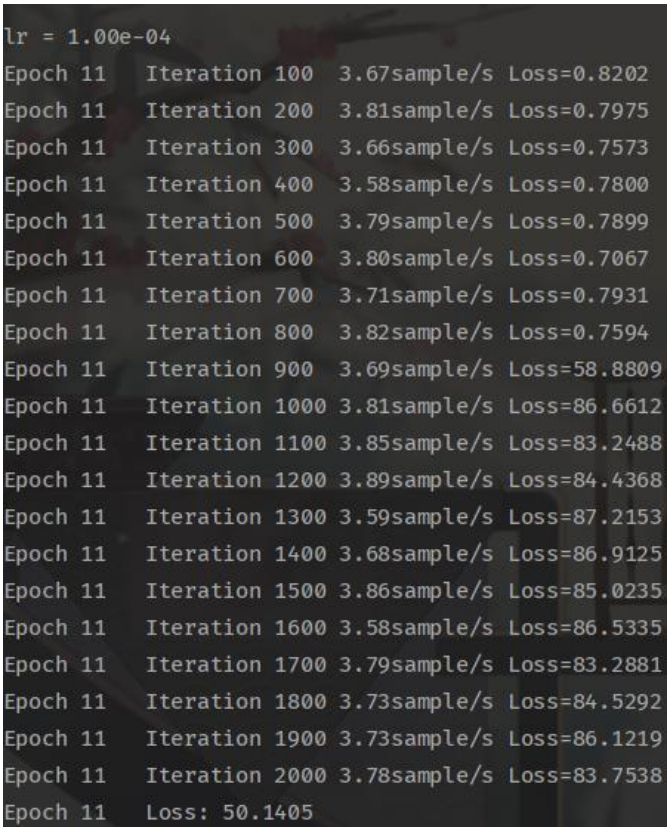

图 4　修改 batch_size 图示

而如果尝试升高版本，第一，前面提到的 convert_params.py 文件无法成功运行，第二，又会遇到另一个问题 version2.1.1 反传失败；Leaf node that doesn't stop gradient can't use inplace operator · Issue #34633 · PaddlePaddle/Paddle (github.com)：在 Tensor 的运算中会出现

stop_gradient 参数不能自适应调整，导致梯度反传失败的情况，如下图所示：

图 5　梯度反传失败图示

目前已经将该问题反馈给 Paddle 官方，希望后续版本能有修复。

六.对 CIE-H 的改进的探索

在阅读 CIE-H 论文时，当我理解了 Hungarian Attention 机制的本质后，我产生了一个想法：Hungarian Attention 机制本质上是将注意力集中于 ground truth 和 permutation matrix 构成的 mask 内，既然如此，我能不能将注意力细分一点，对 ground truth 和 permutation matrix 重合与不重合的区域给予不同的关注，也就是赋予不同的权重？基于这个想法，我进行了如下探索：

我初步的想法是，我们对 ground truth 和 permutation matrix 都为 1 的地方给予权重α，而二者不同的地方给予权重β，一般而言$\alpha < 1$，以使得模型不要轻易陷入局部最优，而$\beta > 1$，以促进模型关注不同的区域，将注意力集中于这些有区别的地方，如下图所示：

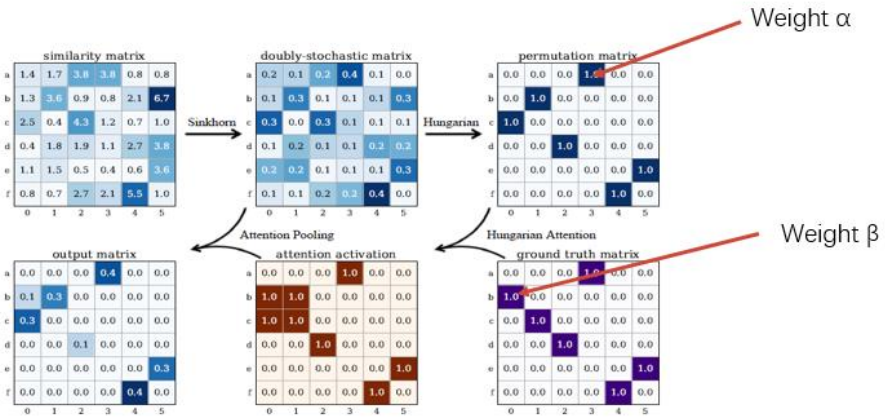

图 6 权重图示

那么这个想法究竟有没有可行性呢？对此我做了一些实验，首先在 willow 数据集上取不同的α和β值进行测试，结果如下表所示：

表 6　在 Willow Object 上训练的平均准确率（%）（30 epoches）

	α=0.1 β=10	**α=0.25 β=4**	**α=0.5 β=2**	**α=0.75 β=1.25**	**α=1 β=1**	**α=2 β=0.5**
Willow	**91.69**	90.73	89.21	87.72	89.46	1.01

可以看到随着α的下降和β的上升，模型的准确率越来越高，说明关注度区分越明显时模型训练效果越好。而当$\alpha > 1$和$\beta < 1$时，训练的 Loss 会飙升，从 50 达到了 100 以上，且越训练越高，而 acc 只有 0.01 左右，这也能说明权重对于模型训练的关注点有着很大的影响。

不过这样修改权重，本质上是在修改对应区域的学习率，那么上面结果的提高，是否是因为调高或调低了权重导致整体学习率的提高或降低而引起的呢？我在这里也做了实验进行探索：

表 7　在 Willow Object 上训练的平均准确率（%）（30 epoches）

	α=0.1 β=0.1	**α=0.1 β=10**	**α=10 β=10**
Willow	9.09	**91.69**	91.48

从表中可以看到，单纯的调低权重，会导致训练崩溃，loss 骤增；单纯的调高权重也即增大整体学习率，效果确实提高了，但是却不及权重有区别时的结果。这说明了两个问题：

1. 为什么 Willow 数据集 benchmark 的结果（88.98）相比于预训练模型的结果（94.47）会低这么多呢？从这个实验可以发现，学习率的大小应该是一个比较重要的原因，提高整体的学习率时准确率上升了许多；
2. 单纯的调低整体的权重，会导致训练崩溃，而如果让α和β有区别时效果却更好，说明了对不同区域给予不同关注度的这个思路是有可行性的。

于是我尝试将这个改动运用在 VOC 数据集上，得到如下的结果：

表 8　在 PascalVOC 上训练的平均准确率（%）（30 epoches）

	α=0.25 β=4	**α=0.5 β=2**	**α=1 β=1**

PascalVOC	65.75	67.80	**68.79**

可以看到这个改动不仅没有提高，反而让准确率下降了很多，这让我感到非常不解，于是我又探究了一下整体改动的影响：

表 9　在 Willow Object 上训练的平均准确率（%）（30 epoches）

	α=0.25 **β=0.25**	**α=0.5** **β=0.5**	**α=2** **β=2**	**α=4** **β=4**
PascalVOC	——	**69.35**	19.91	——

我发现在$\alpha = 0.25$、$\beta = 0.25$，$\alpha = 2$、$\beta = 2$和$\alpha = 4$、$\beta = 4$的时候训练都崩溃了，而当$\alpha = 0.5$、$\beta = 0.5$时准确率较原来有一定提升。我详细的研究了每种参数对应的 loss 函数的变化过程，发现当权重较大时 loss 会在最初的几个 epoch 骤降，而后面的 epoch 反而越训练越高，准确率越来越低，于是得出一个初步的结论：对于 voc 这样较大较难的数据集，训练时本来需要较多的 epoch 进行拟合，过度提高部分区域的权重反而会让模型在最开始的训练快速收敛而陷入局部最优。相反，降低整体权重（即调节学习率）反而能让模型在更多的 epoch 中稳定的学习。

基于此分析，我萌生了一个想法：将α和β设置为固定值可能并不是一个很好的选择，更好的设置方式应该是一个与 epoch 有关的函数，比如让早期的 epoch 更多关注不一样的区域，而训练到后面适当回调α和β的值，那么这样会不会有效呢？我探索了一种线性的变化，即α和β从初始值线性变化到 1，结果如下：

表 10　在 Willow 和 PascalVOC 上训练的平均准确率（%）（30 epoches）

	α=0.2 **β=5**	**α=0.5** **β=2**	**α=0.75** **β=1.25**	**α=1** **β=1**
Willow	**91.89**	89.43	87.58	89.46
PascalVOC	66.50	68.48	68.51	**68.79**

可以看到这样的改进相较于之前有一定的提升，说明权重的线性变化是有效果的。

另外，我在初次实现这个改动的时候实现的方式有误，将α和β直接乘在了 predicted matrix 和 ground truth 上，结果跑出来的结果反而更好，如下：

表 11　在 Willow 训练的平均准确率（%）（30 epoches）

	α=0.25 β=4	**α=0.5 β=4**	**α=1 β=1**
Willow	**91.96**	91.90	89.46

分析原因，我认为这样的改动本质上是拉大了 predicted matrix 和 ground truth 之间的距离，其实质与上面调整权重是一样的，而后我在此之上尝试了权重线性变换，结果有小幅提升。

最后，我也尝试了在 Paddle 版本中利用这种方法，这里选取$\alpha = 0.5$，$\beta = 2$：

表 12　在 Willow Object，PascalVOC 和 CUB2011 上训练的平均准确率（%）（Paddle）

model	CIE-H Change	CIE-H Paddle	CIE-H Torch
Willow	79.19	79.47	**89.46**
PascalVOC	43.87	48.99	**68.79**
CUB2011	73.65	75.51	**94.87**

(注：CIE-H Change 表示改进的 CIE 版本)

可以看到这个改动的效果并不理想，不过我觉得这个是正常的，因为目前 Paddle 训练过程的 loss 回传有问题。

总结一下我的探索过程：

1. 分析了调节权重给训练带来的影响：调节不同区域的权重在小数据集（如 willow）上会带来提升，而对于大数据集（如 voc）则会起反效果。
2. 尝试使用线性变化的权重进行训练，并取得了一定的成果；
3. 探究了另一种可能的改动方式，并取得了较好地效果；
4. 在 Paddle 版本中实现了该改动。

最后列出我目前达到的三个数据集上效果最好的三种情况：

表 13　在 Willow Object，PascalVOC 和 CUB2011 上训练的平均准确率（%）

model	CIE-H Change	CIE-H Torch	method	**α**	**β**
Willow	**92.03**	89.46	modified2	0.25->1	4->1

PascalVOC	**69.35**	68.79	modified1	0.5	0.5
CUB2011	**95.06**	94.87	modified2	0.5	2

(注：CIE-H Change 表示改进的 CIE 版本)

七.总结

本次实验，我阅读了图匹配方向的多篇论文，对该领域有了大致的了解，随后我精读了 CIE-H 这篇论文，详细研究了其机制并将自己的一些想法用于改进该机制。与此同时，我配置环境复现了 PyTorch 的测试、训练功能以及 Paddle 的测试功能。我作出的额外贡献主要有三点：

1. 利用 CIE-H 在 CUB2011 数据集上实现训练功能；
2. 实现了 Paddle 的训练功能，并分析了结果不理想的原因；
3. 对 Hungarian Attention 机制尝试一些新想法，并进行了大量的实验和探索。

参考文献

[1] T. Yu, R. Wang, J. Yan, and B. Li, "LEARNING DEEP GRAPH MATCHING VIA CHANNEL- INDEPENDENT EMBEDDING AND HUNGARIAN ATTEN- TION," in *ICLR2020*, 2020.

[2] E. Çela, "The Quadratic Assignment Problem," *INFORMS,* 1963.

[3] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," 2005.

[4] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *European conference on Computer vision*, 2010: Springer, pp. 492-505.

[5] F. Bernard, C. Theobalt, and M. Moeller, "Tighter Lifting-Free Convex Relaxations for Quadratic Matching Problems Download PDF."

[6] A. Zanfir and C. Sminchisescu, "Deep learning of graph matching," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2684-2693.

[7] R. Wang, J. Yan, and X. Yang, "Learning combinatorial embedding networks for deep graph matching," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3056-3065.

[8] Z. Zhang and W. S. Lee, "Deep graphical feature learning for the feature matching problem," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5087-5096.

[9] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903,* 2017.

[10] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998-6008.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*, 2017: PMLR, pp. 1263-1272.

[12] T. Yu, J. Yan, Y. Wang, W. Liu, and B. Li, "Generalizing graph matching beyond quadratic assignment model," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 861-871.