2021-2022 春季学期创新实践 II 第二次答辩报告

面向旅行商问题的神经网络求解器复现及泛化性能测试

吕昊泽 11912814

张子研 11912324

指导老师： 唐珂教授 刘晟材老师

2022. 4. 29

目录

1 项目介绍

1.1 前提描述

在开题报告中，我们介绍了神经网络求解器求解旅行商问题的技术背景，并对该领域开创性的模型 AM[5] 进行了性能和泛化性测试。所谓神经网络求解器求解 TSP 问题，就是使用 Encoder-Decoder 结构，用 Encoder 对 TSP 问题进行信息提取后用 Decoder 基于注意力机制进行节点选择概率的生成，最终构造完整的 TSP 问题的解。而其根据的 Encoder-Decoder 模型最早见于 seq2seq 模型，而近年来对于 TSP 神经网络求解器的诸多研究，都局限于该框架中。人们通过使用不同的技术替换 Encoder，Decoder 以及模型的训练方法，来获得性能上的提升。现阶段，在该领域表现突出的架构主要有 GCN 和 Transformer 两种。

1.2 项目进展

在第二阶段的学习中，我们选取了更多的 TSP 神经网路求解器进行了性能测试，对基于 GCN 技术的 TSP 神经网络求解器性能有了基本了解。与此同时，我们基于 AM 模型的测试思想，构建了通用测试数据集，用于测试不同的 TSP 神经网络求解器的求解性能和泛化性能。最后，我们对于位置编码进行了简单的探究。

1.3 报告结构

在本篇报告中，在第二部分简介 GCN_NPEC[1]模型和 GPN[2]模型，在第三部分介绍两模型求解性能测试，在第四部分简单总结现阶段位置编码的方式并进行了简单的探究， 在第五部分将进行总结和进一步规划。

2 模型简介

2.1 模型背景知识简介

2.1.1 LSTM/GRU

LSTM 与 GRU 的存在是为了解决简单 RNN 面临的长期依赖问题（由于反向传播存在的梯度消失或爆炸问题，简单 RNN 很难建模长距离的依赖关系），一种比较有效的方案是在 RNN 基础上引入门控机制来控制信息的传播。

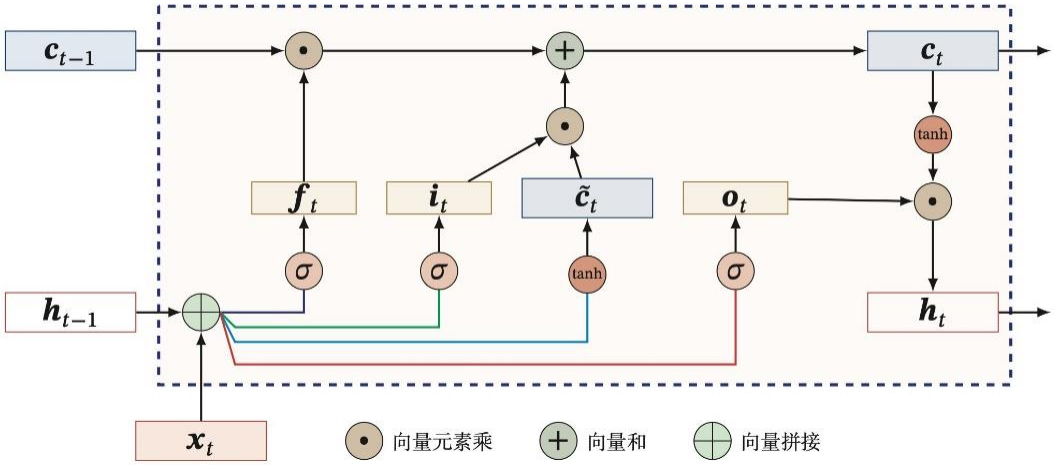
① LSTM

LSTM 通过引入了三个门来控制信息的传递，分别是遗忘门f_t，输入门i_t和输出门o_t。三个门的作用为：

（1） 遗忘门f_t控制上一时刻的内部状态c_{t-1}需要遗忘多少信息；

（2） 输入门i_t控制当前时刻的候选状态$\check{c}_t$有多少信息需要保存；

(3) 输出门o_t控制当前时刻的内部状态c_t有多少信息需要输出给外部状态h_t；



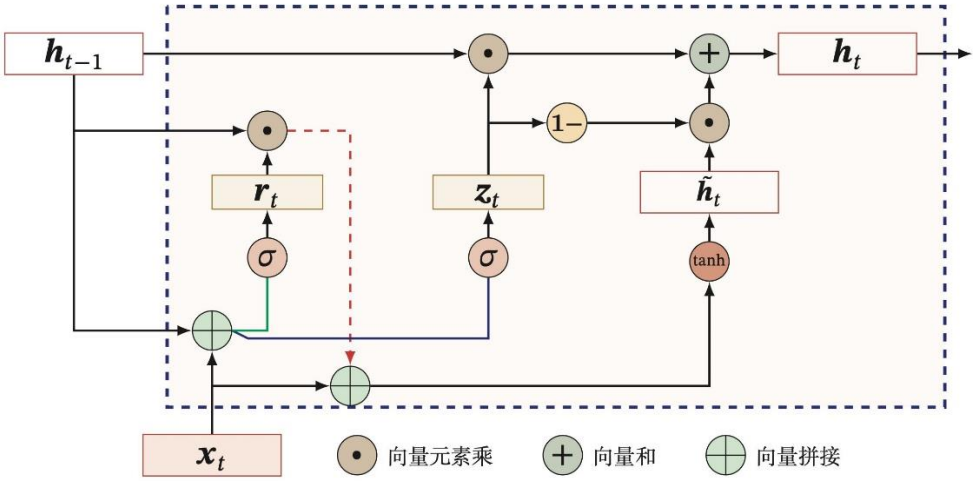
图表 1 展示了 LSTM 的工作原理[6]

② GRU

GRU 的结构比 LSTM 更为简单一些，GRU 只有两个门，更新门z_t和重置门r_t。

(1)更新门z_t：控制当前状态h_t需要从上一时刻状态h_{t-1}中保留多少信息（不经过非线性变换），以及需要从候选状态$\check{h}_t$中接受多少信息；

(2)重置门r_t：用来控制候选状态$\check{h}_t$的计算是否依赖上一时刻状态h_{t-1}



图表 2 展示了 GRU 的工作原理[6]

由于 TSP 神经网络求解器的 Decoder 模型大多是按照时间顺序依次输出 TSP 问题的节点选择概率，以此得到 TSP 问题的解的。因此他们大多使用了 RNN 模型框架，因此 LSTM 和 GRU 在 TSP 神经网络求解器中都有较为广泛的使用：GCN_NPEC 使用了 GRU 而 GPN 模型使用了 LSTM。

2.1.2 GCN

近年来，人们对深度学习方法在图上的扩展越来越感兴趣。在多方面因素的成功推动下，研究人员借鉴了卷积网络，循环网络和深度自动编码器的思想，定义和设计了用于处理图数据的神经网络，由此一个新的研究热点“图神经网络（Graph Neural Network GNN）”应运而生。

图神经网络分为很多种，分别是图卷积网络（GCN）、 图注意力网络（GAT）、图自编码器（ Graph Autoencoders）、图生成网络（ Graph Generative Networks）和图时空网络（Graph Spatial-temporal Networks）。在 TSP 神经网络求解器中应用广泛的主要是带注意力机制的图卷积网络(一般在 TSP 神经网路求解器中被称为 GCN，但是在上述分类中可以认为是 GAT)。

GCN 主要分为两种方法，分别是谱方法和空间方法。谱方法通过傅里叶变换把从图信号处理的角度引入滤波器来定义图卷积，其中卷积认为是从傅里叶变换产生的图信号中去除噪声，而空间方法则是将卷积表示为从邻域聚合信息。TSP 神经网络求解器中所使用的 GCN 基于空间方法，可以认为对于图中的每一个点，都通过某种方法确定其在图中的邻居是什么，再通过注意力机制有选择的从它的邻居中进行信息的聚合。

2.2 模型简介和特点

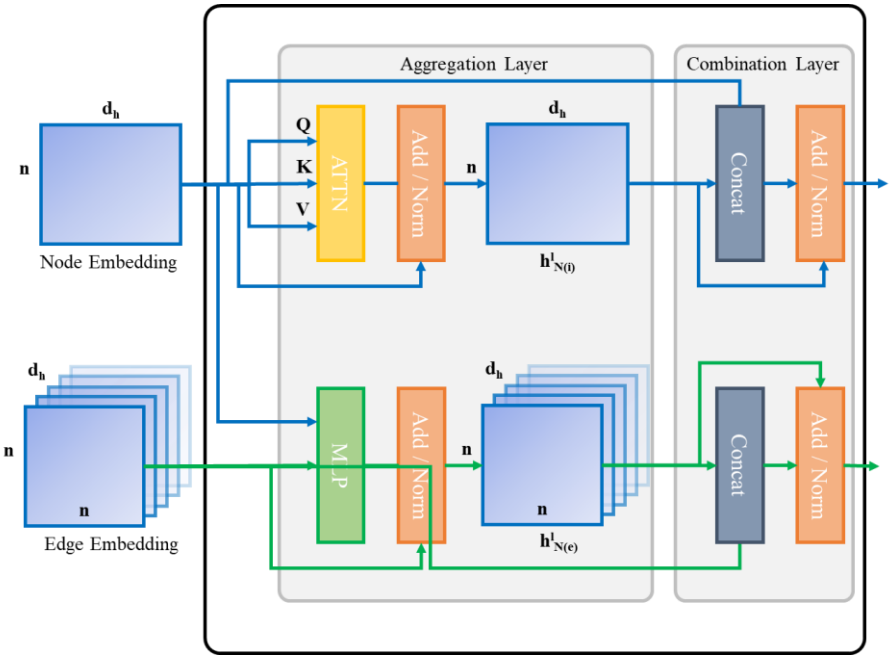
2.2.1 GCN_NPEC 模型架构简介

GCN_NPEC 模型是由阿里巴巴和菜鸟团队提出的一个用于解决 CVRP 问题的模型，其结构依然采用 Encoder-Decoder 模式，使用的创新技术技术包括 GCN，联合学习等。

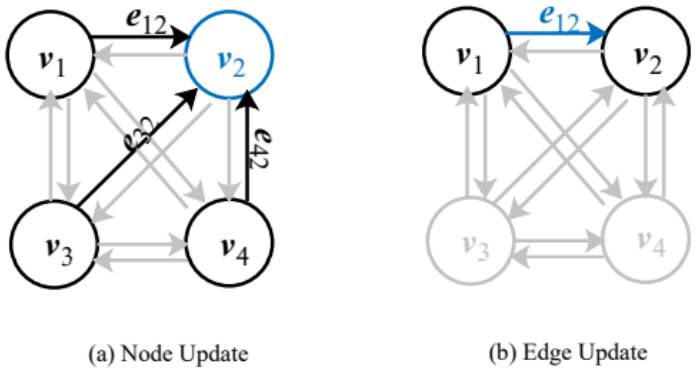
① Encoder

如图表 3 所示，GCN_NPEC 模型的编码器是由 L 层 GCN 层组成的。由于其求解的是 CVRP 问题，因此输入包括节点坐标，节点需求，边距离矩阵。这里节点坐标和节点需求被处理成点嵌入，而边距离矩阵被处理成边嵌入。点和边的嵌入经过 Encoder 的编码生成 Encoder embedding，同样分为边和点两部分。

每层卷积层由聚合子层和组合子层两部分组成。如图表 4 所示，在聚合子层中，GCN 对各个节点和边做信息聚合。在组合子层中，来自聚合子层的聚合嵌入，将会和节点，边的初始输入通过 concat 方式进行组合，生成新的节点，边嵌入。此外，GCN 中的每一层都包括一个残差连接和一个 LayerNorm 操作。



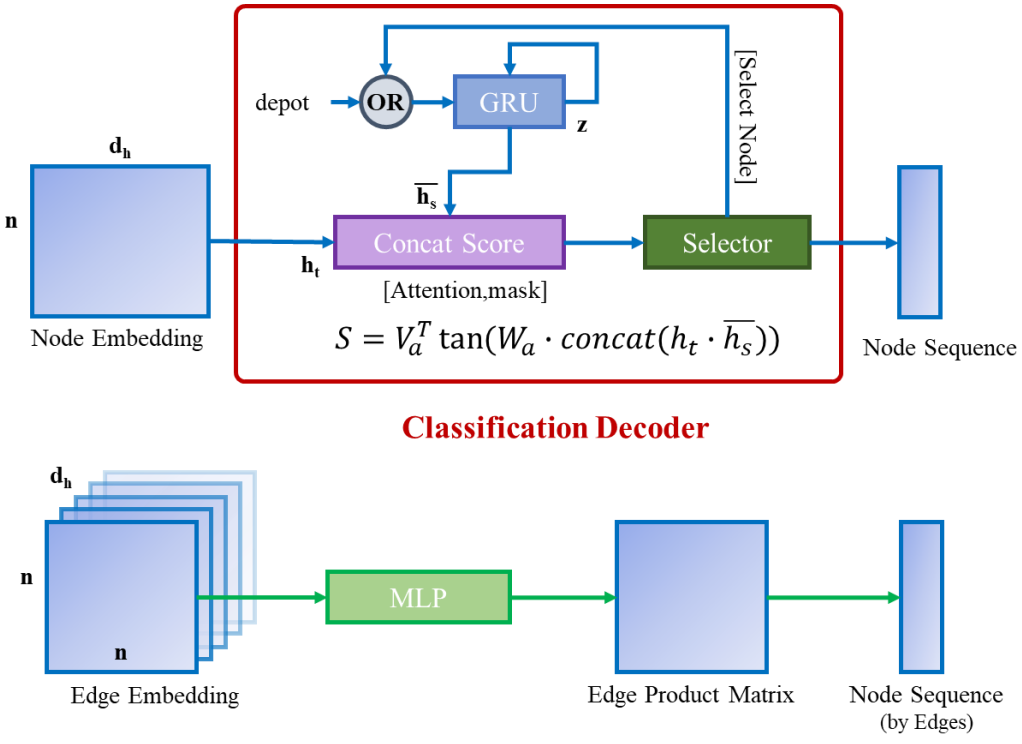
图表 3 GCN_NPEC Encoder 架构示意图



图表 4 GCN_NPEC 模型点，边信息汇聚实例[1]

② Decoder

与常见 Decoder 架构不同的是，GCN_NPEC 模型使用两个解码分别对边嵌入和点嵌入进行解码，生成各自的解。



图表 5 GCN_NPEC 模型 Decoder 架构示意图

如图表 5 所示，序列解码器接受节点嵌入，结构基本和 15 年的指针网络所用结构相同，在每一个时间步，用 GRU 生成对当前时刻的信息嵌入，用当前时刻的信息嵌入和编码器点嵌入一起，使用 seq2seq 中的 concat 注意力机制生成输入的概率，通过选择器产生解。值得注意的是，这里使用了掩码机制，屏蔽掉了那些不符合问题规则的概率。

分类解码器则是接受边嵌入信息，通过 MLP 生成边选择概率矩阵，通过概率矩阵选择边生成解

③训练方法

GCN_NPEC 模型训练策略将监督学习和强化学习相结合。序列解码器使用 TSP 神经网络求解器常见的训练方法“强化学习”进行训练。采取的具体策略与 AM 模型的 Rollout 方法相同。而分类解码器则使用监督学习进行训练，训练使用的 Label 由序列解码器产生。最终模型损失由两个解码器的损失之和计算得出。

2.2.2 GCN_NPEC 模型创新

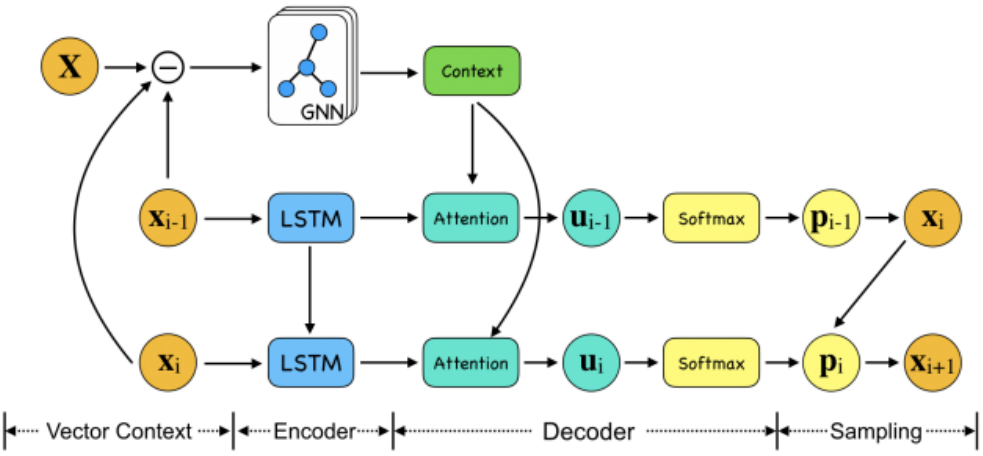
① 在神经网络求解器中，第一次将节点特征和边特征共同考虑，而之前模型仅仅考虑欧氏距离，与现实应用场景有较大差别。GCN_NPEC 使用基于真实数据与采样的训练集进行训练，并在物流平台真实数据上进行了测试，得到了较好的表现结果。

② 由于同时使用了节点数据和边数据，因此 GCN_NPEC 使用了两种编码器分别对点嵌入和边嵌入进行解码

③ 由于使用的两个解码器训练方法不同，GCN_NPEC 模型创新型的使用联合学习的策略，将强化学习和监督学习的解码器共同训练。

2.2.3 GPN 模型架构简介

GPN 为哥伦比亚大学大学团队提出的 TSP 神经网络求解器。该求解器遵循 Encoder-Decoder 架构，使用 LSTM 和 GNN 作为 Encoder，使用注意力机制作为解码器。此外，该模型还提出了分层强化学习，以针对复杂问题的求解。



图表 6 GPN 模型架构示意图[2]

① Encoder

GPN 使用 GNN 和 LSTM 共同组成 Encoder。其中 GNN 负责对整个图进行信息的抽取，抽取方式依然是根据注意力机制从周围的节点中做信息聚合。但是这里 GNN 的输入与前人不同，它通过前一时刻的输出对输入进行了额外的处理，把之前由坐标生成的 embedding 变成了指向前一节点的向量生成的 embedding。

$$X^l = \gamma x_i^{l-1}\theta + (1-\gamma)\phi_\theta\left(\frac{1}{N(i)}\ \{X_j^{l-1}\}\right) j \in N(u)\ \cup\ (i)$$

② Decoder

解码器接受来自 LSTM 的隐藏态和来自 GNN 的 graph embedding。使用注意力机制，将 LSTM 的隐状态作为 q， graph embedding 的各个向量作为 r 使用注意力机制对各个节点生成选择概率。

$$u_i^{\{j\}} = v^T * \tanh\left(W_r r_j + W_q q\right)$$

最后通过 softmax 生成节点选择概率。

③多层强化学习

作者为了解决带有约束的 TSP 问题，设计了多层强化学习模型来适应约束条件。具体思路为在每一层中都对于约束中的某一特定问题进行优化，从下到上解的范围逐渐变小，逐渐满足约束，从而在最上层输出满足优化的解。但是本学期我们研究的是旅行商问题，且由于这部分模型训练可能需要耗时几天，因此本阶段还没有做更多详细的研究。

2.2.4 GPN 模型创新

① 该神经网络首次提出了多层强化学习，通过将约束项拆分解决的思路来解决有约束的组合优化问题。

② 该模型提出了 vector context，不使用之前 GNN 一直使用的 node embedding 作为输入，反而使用 vector context，也许能代来更多的坐标信息，并且在更大规模的 TSP 图中表现更好。

3 论文相关实验和总结

根据第一阶段对于 AM 模型的研究，参考 AM 模型论文的实验方法，从求解性能，训练收敛性能，泛化性能和训练效率四方面对比 3 个模型的性能，补全了第一阶段的数据的同时进行了总结。

3.1 TSP 实验结果

本实验中使用的设备的设备如表格 1。

设备	本实验
CPU	Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz
GPU	TITAN RTX(24G)

表格 1 本实验所用设备

3.1.1 求解性能实验

① 概述

该实验中对比了随机生成的 TSP 问题下 AM,GCN_NPEC,GPN 模型的表现(时间尺度，结果尺度)。

② 实验结果

对于 AM 模型，我使用了论文中提供的与训练模型和自身训练的模型进行测试。

对于 GCN_NPEC 模型，我是用了自身训练的模型进行测试，但是由于其发表在 KDD(2020)上，并未提供任何可参考代码和他人复现代码，所以我没有复现出论文中描述的结果，我认为这与论文对于 Decoder 部分描述不清有关。这里列出我自身训练的 GCN_NPEC 模型结果。

对于 GPN 模型，由于它未提供训练好的模型，更改其代码后进行训练，使用

自身训练好的模型进行测试。

	n=20			n=50			n=100		
Method	Obj	Gap	Time	Obj	Gap	Time	Obj	Gap	Time
Concorde	3.84	0%	6m	5.70	0.00%	12m	7.76	0.00%	3m
LKH3	3.84	0%	2m						
AM(greedy)	3.84	0%	1s	5.79	1.57%	3s	8.10	4.38%	8s
AM(sample)	3.84	0%	14m						
AM(self,greedy TSP20)	3.84	0%	1s	5.97	4.73%	3s	8.97	15.59%	8s
GPN(self, greedy, TSP20)	3.88		10s	8.51		20s	15.45		1m
GPN(self, greedy, TSP50)	3.87		1s	5.79	1.57%	3s	8.11		8s
GPN(self, greedy, TSP50)									
GPN(self, greedy, TSP50, vector_context)	4.03		15s	6.06		30s	8.53		1m
GCN_NPEC(self,greedy, CVRP20)	5.03	30.98%	1m	10.784		2m	21.216		6m

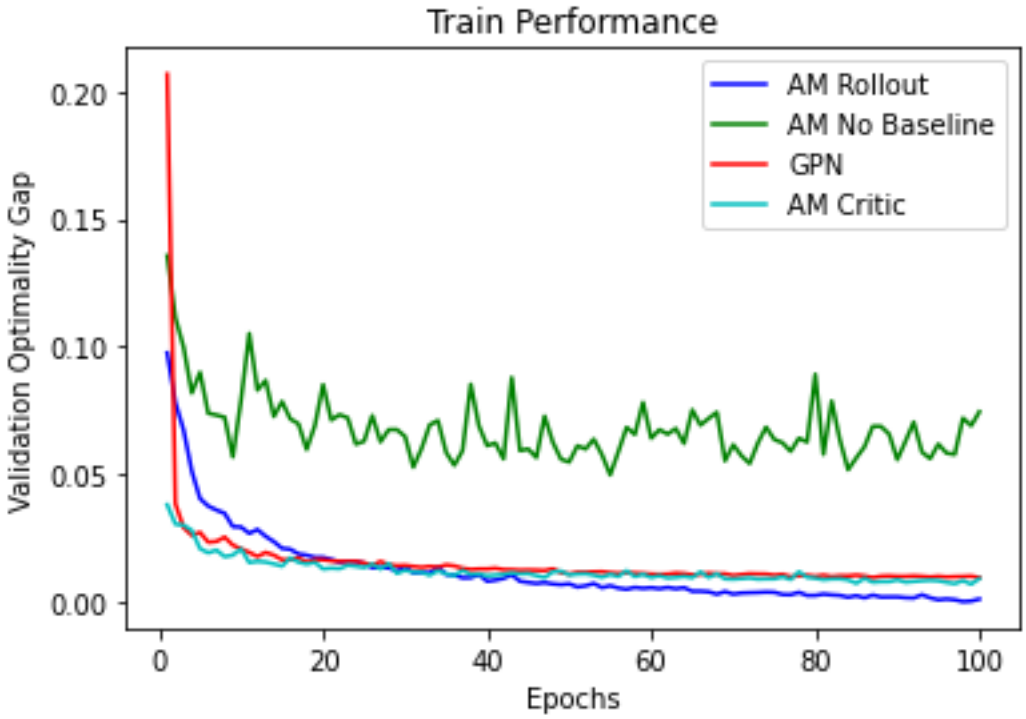
表格 2 使用 AM, GCN_NPEC, GPN 对比传统模型(Baseline)。其中 Obj 为模型最优解，Gap 为模型最优解和历史最优解的差距(百分率)

从表格 2 中可以看到，AM 模型在 TSP20, 50, 100 3 个数据集上的表现都要优于其他竞争对手。GPN 模型在未使用 vector context 时，在训练数据集大小相同的测试数据集上表现较好，但是似乎失去了泛化性能。而使用 vector context 后，GPN 表现出了较好的泛化性能，虽然与 AM 模型尚有差距，但是考虑到其只训练了 10 epoch，性能值得肯定。而我复现的 GCN_NPEC 数据为使用 CVRP20 数据集训练 75 epoch 的模型，怀疑模型还在欠拟合状态，但是该模型训练过慢且收敛不稳定，所以没有继续进行。可以看到 GCN_NPEC 模型的求解效果和求解时间均不理想。

3.1.2 模型收敛性能测试

① 概述

在该实验中，以图片的形式展现了 AM, GPN 模型随着训练轮次的增加在测试机数据上解的变化。没有 GCN_NPEC 模型是因为其训练是采用的是 CVRP 数据集，且其训练不稳定，无法复现同样的训练过程，故不参与实验。



图表 7 不同 Baseline，模型的结合的训练效果

② 实验结果

如图 7，可以看到基于 Rollout 方法作为 Baseline 的 AM 模型在全局表现出了最好的模型训练效果。而使用 Central Self-Critic(rollout 方法的变种)的 GPN 模型在训练前期表现出了最好的训练效果，在 20 epoch 之前模型表现优于 AM 模型。使用 critic 作为 Baseline 的 AM 模型表现出了与 GPN 模型相似的训练能力，均是在前期的训练效果由于 Rollout，但是在 20 epoch 后被 Rollout 超过，并无法达到 Rollout 最终的训练效果。与它们作为对比的没有使用 Baseline(估计值恒为 0)的模型训练表现相对较差，始终无法稳定收敛。

该实验证明了 Baseline 方法的有效性，并且展现了三种不同 Baseline 方法在训练效果上的差异。

3.1.3 模型训练时间实验

① 概述

实验对比了不同模型在不同数据集下的训练时间和最终推断结果，作为模型训练时间的参考。。

② 自身实验

模型	数据集/设备	Epoch Time	Learning Rate $\{10^{-3}\} * 0.96^{epoch}$
Seed = 1234			
AM	TSP20(RTX 2080Ti)	4:26(batch size 512)	3.85(0.26%)
AM	TSP20(RTX TITAN)	6:53(batch size 512)	5.79(1.66%)
AM	TSP100(RTX TITAN)	24:00(batch size 256)	
GPN	TSP20(RTX TITAN)	4:00(batch size 512)	3.87(0.78%)
GPN	TSP50(RTX TITAN)	24:45(batch size 512)	
GCN_NPEC	CVRP20(RTX TITAN)	4:16(batch size 256)	5.04(30.98%)

表格 3 使用不同的模型和训练集下模型训练速度和最终效果

如表格 3，在 RTX TITAN 训练下，AM 模型表现出了总体趋势上最小的训练时长，并且训练出来的模型在 TSP 问题上求解的性能也是最佳的。GPN 和 GCN_NPEC 模型虽然在 TSP20 数据集上的训练时间低于 AM 模型，但是 GPN 随着数据集的增大训练时间大幅度增加，推测与其 GNN 需要运行 n(节点总数)有关。而 GCN_NPEC 模型在我测试中训练效果还是远差于 AM 模型，难以收敛，处于欠拟合状态。

3.1.4 模型的在不同大小问题下的泛化性能测试

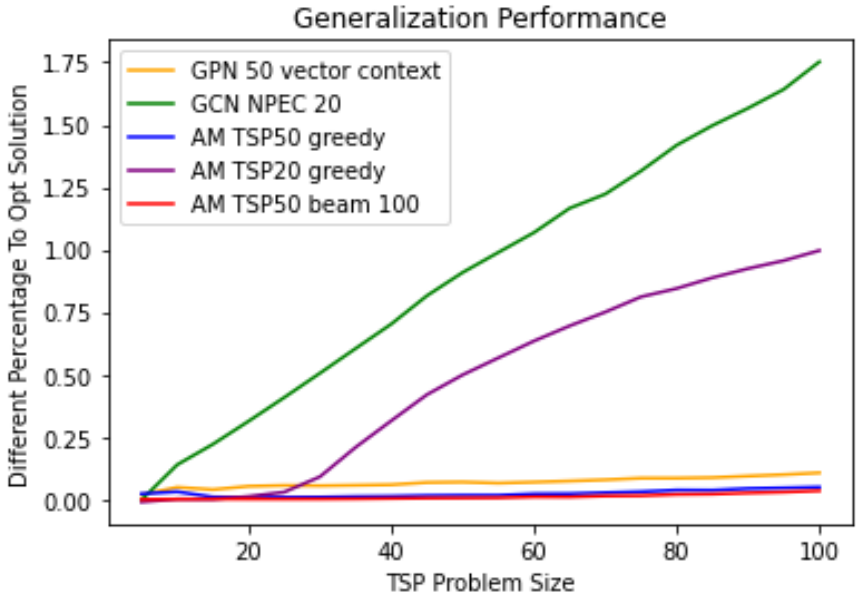
① 概述

实验对比了不同模型在不同问题大小上泛化程度。如将用 Graph size = 20 训练的模型去解决 size = 100 的问题，看其泛化性如何。实验测试了 Graph size = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 75, 80, 85, 90, 95, 100] 上的泛化性能表现。

② 自身实验

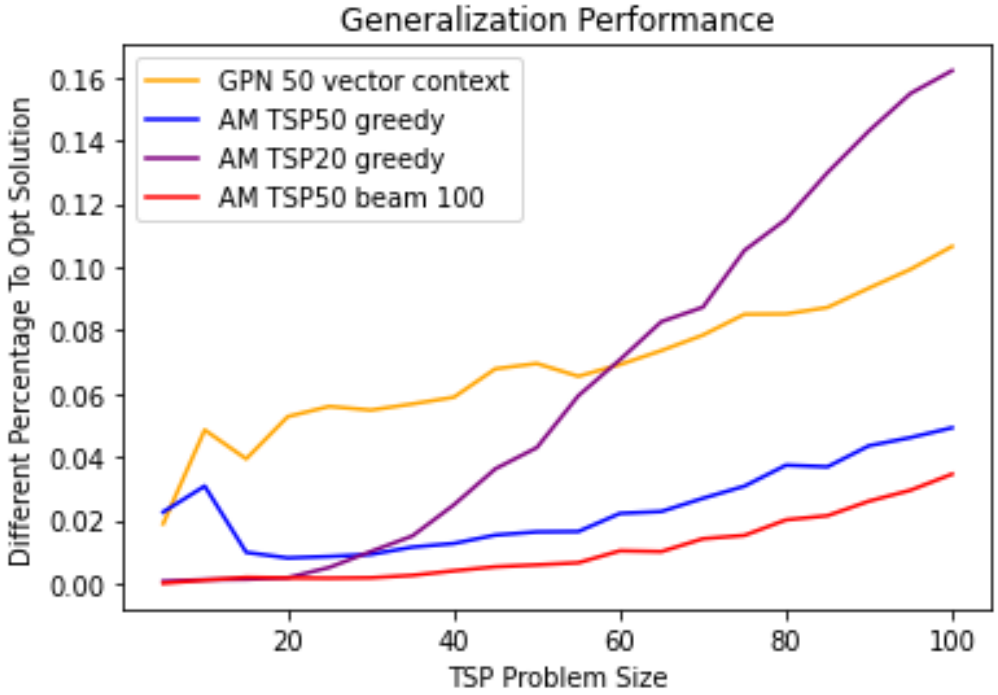
本实验选取了之前实验中泛化性能较好的 AM 50 sample 和 AM 50 greedy 作为 AM 模型的代表，与 GCN_NPEC 20, GPN 20, GPN 50 context 作泛化性能测试。首先通过模型自带的数据生成函数生成数据的对应数据集，其中每个数据集包含个测试样例。将数据转化成 TSPLIB 数据格式使用 Concorde[7]求解器求解出各个数据集的解，每个数据集取平均作为 Opt Solution。后分别使用模型预

训练好的模型去求解对应数据集，将得到的结果与 Opt Solution 进行比较，获得泛化性能曲线。



图表 8 不同模型在不同大小数据集上的泛化性能测试

在图表 8 中可以看到，GCN_NPEC 模型和 GPN_20 模型的泛化性能与其他模型差距较大，前者可能是由于训练不稳定，欠拟合的原因，后者则是在训练书积极附近能保持稳定的性能，但一旦超过一定值，解的质量就大幅度下降。因此将它们去除，添加 AM 20 greedy 生成新的图。



图表 9 不同模型在不同大小数据集上的泛化性能测试(优秀模型精细对比)

在图表 9 中可以看到，GPN 模型在使用了 vector context 后泛化性能有了明显的提升，与 AM 模型的差距大幅度减少，甚至在数据集大小大于 60 时超过了 AM 20 greedy。考虑到其是一个仅仅训练了 10 轮的模型，其泛化性能值得肯定。而毫无疑问，AM 50 sample 模型依然以极强的泛化性能拔得头筹，与 Baseline 的差距始终在 5%以内。

3.2 论文实验复现总结

在创新实践的第二阶段，我根据前一阶段对 AM 模型所做实验的理解，对 GCN_NPEC, AM 和 GPN 模型进行了测试，从求解性能，训练收敛性能，泛化性能和训练效率四方面对比了 3 个模型。

首先，AM 模型在全部四项测试中都是整体表现最为优异的一个模型，其在解的质量和训练时间上都领先于其他模型。而其提出的 Rollout Baseline 也被后续其他模型借鉴并使用。

GPN 模型在求解质量上略逊于 AM 模型，而在泛化性能上，未使用 vector context 技术的 GPN 模型在泛化性能上与 AM 模型相差甚远，但是使用 vector context 技术后，GPN 模型的泛化性能产生了数量级级别的提升，而其更少的训练轮数也让其相对较弱的泛化性能与训练花费产生权衡。

而 GCN_NPEC 模型，由于其代码未开源的原因，且论文中对于 Decoder 的描述过于模糊，导致我未能复现出论文展示出来的求解性能。其在训练上同样存在，收敛过慢，稳定性不足等缺点。

但是必须要提到的是，GCN_NPEC 模型和 GPN 模型都针对其他问题进行了额外的优化，并在其论文实验中，他们的特化部分均获得了超越 AM 模型的表现。但是由于本实验着眼于传统 TSP 问题，且其他问题的训练时间过长，因此在本轮实验中没有进行相应对比。

4 位置编码相关

4.1 位置编码方式

通过对于 Vision Transformer[3]的阅读，我了解到了现阶段位置编码的方式主要分为以下四种。

① 人为设计的数学公式 最为出名的便是 Transformer[4]中所用的 Sinusoidal 位置编码.通过认为设计的三角函数与幂函数的结合，实现相对位置越近的两词位置编码的余弦相似度越近。

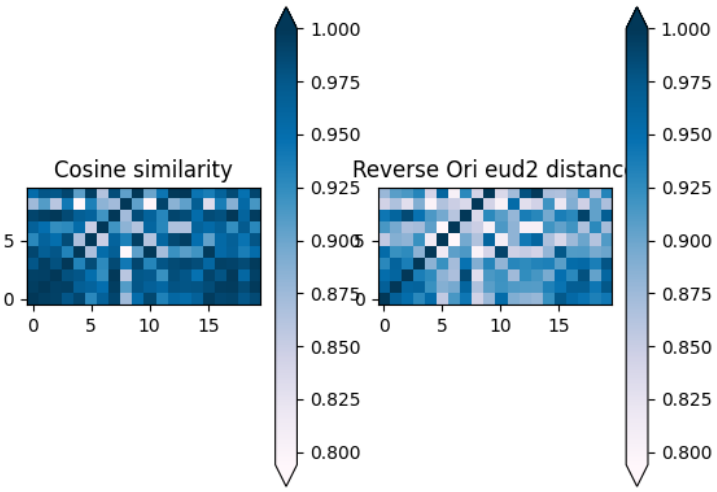
② 1-D position embedding 认为输入是一组按照顺序排列的栅格行。可以不认为指定参数，而是通过一个 NLP 的方式形式工作，VIT 中通过余弦相似度验证了该编码方式在图片位置编码上的有效性。

③ 2-D positional embedding 考虑输入是一个二维网格网络。VIT 中对其的处理是学习两个 position embedding，分别对比行和列，然后基于输入的坐标，连接行列嵌入，获得最终的位置嵌入。VIT 实验中该编码方式与 1-D 编码方式并无性能上的区别。

④ Relative positional embedding 通过一维注意力定义所有输入之间的相对位置。在运算时，将基于位置的注意力编码添加到基于内容的注意力编码中后再进行 softmax 运算，来完成相对位置编码的嵌入。

4.2 AM 位置编码实验

根据 VIT 论文的总结，我们可以认为 AM 模型使用的便是 Learnable 1-D position embedding 方式。而根据 VIT 论文的是实验，该编码方式是有可能学到 2D 相对位置关系的，那么事实是否真的如此，我仿照 VIT 论文进行地实验，使用余弦相似度衡量各个节点 position embedding 的相似度，并于它们之间标准化的 EUD2 距离进行对比。



图表 10 位置编码的余弦相似度和表转化的 EUD2 距离对比

可以看到，各个节点的 position embedding 和其标准化的欧氏距离之间存在数值相对大小上的相似性。欧氏距离相对较大的地方，其 position embedding 之间的余弦相似度也同样较大。但是在绝对数值上，依然有不小的差距，在一些地方，节点余弦相似度之间的差距远小于它们欧氏距离大小上的差距。

该实验证明了 AM 模型使用的 position embedding 方法能够学到大部分欧氏距离所表现出来的空间信息，但是由于其方法是基于 Learnable 1-D position embedding，因此可能其模拟性能存在性能瓶颈。这也许能解释为什么 GPN 模型添加了 vector context 之后泛化性能会有大幅度提升。猜测是因为 GPN 模型原来使用的 Learnable 1-D position embedding，在节点较多的情况下已经不能提取位置信息，因此通过认为引入向量位置信息，减轻了 position embedding 的压力，从而表现出了更好的泛化性。

5 现阶段总结与规划

5.1 现阶段总结

在创新实践的第二阶段，我们根据第一阶段对 AM 模型的实验经验，构建实验从四个方面评估了 AM, GCN_NPEC, GPN 三个模型的性能，并总结了它们的性能优异，模型特点和不足。此外，我们还从 VIT 论文中得到启示，对 position embedding 的方法和其有效性进行了简单的实验。

5.2 下一阶段安排

下一阶段，我们预计还将引入更多的模型进行性能测试，同时参考更多文献，对我们的测试方案进行进一步的完善。

6 引用

[1] Ma Q, Ge S, He D, et al. **Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning.** In: *arXiv:1911.04936, 2019.*
[2] Duan, L., Zhan, Y., Hu, H., Gong, Y., Wei, J., Zhang, X., & Xu, Y. (2020, August). **Efficiently solving the practical vehicle routing problem: A novel joint learning approach.** In: international conference on knowledge discovery & data mining, ACM SIGKDD, 2020

[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). **An image is worth 16x16 words: Transformers for image recognition at scale.** In: *arXiv:2010.11929.*

[4] Vaswani A, Shazeer N, Parmar N, et al. **Attention is all you need.** In: *Advances in Neural Information Processing Systems, NIPS, 2017.*

[5] Kool W, Van Hoof H, Welling M. **Attention, learn to solve routing problems!** In: *7th International Conference on Learning Representations, ICLR, 2019*

[6] **LSTM 与 GRU 的原理**. (2022). Retrieved 28 April 2022, from *https://zhuanlan.zhihu.com/p/184937263*

[7] **Concorde Home**. (2022). Retrieved 24 March 2022, from *https://www.math.uwaterloo.ca/tsp/concorde.html*

[8] **LKH-3** (Keld Helsgaun). (2022). Retrieved 24 March 2022, from *http://webhotel4.ruc.dk/~keld/research/LKH-3/*