2021-2022 春季学期创新实践 II 第三次答辩报告

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 项目背景

近年来，随着 Transformer 等一系列模型带来的注意力机制的飞速发展，以及 Encoder-Decoder 结构为基础的模型的大量提出，神经网络求解器求解组合优化问题逐步成为现实。而旅行商问题作为组合优化问题的经典问题，也受到了人们的关注。从 2015 年开始，随着指针网络的提出，基于旅行商问题的神经网络求解器得到了飞速发展。这学期，我们将以该领域经典模型为基础，通过复现论文的方式，总结一套完备的对模型进行性能测试的方法，并尝试在训练数据上做出一些进展。

1.2 项目进展

在第二阶段，通过对 AM, GPN 和 GCN_NPEC 模型的复现，我们总结出了一套相对完备的模型性能测试方法。在第三阶段，我们着眼于训练数据，根据论文[9]生成了带有明显视觉特征的 TSP 数据集，并使用它们分别训练了该领域的经典模型，并将我们之前总结出来的测试方法应用在它们身上，测出它们的泛化性能和求解性能。而面对不符合直觉的结果，我们参考相关论文，设计实验对数据集，平均注意力距离和位置编码进行分析，尝试对产生的结果进行了解释。

1.3 报告结构

在本篇报告中，在第二部分将使用 python 复现论文[9]的多个变异算子，生成对应的 TSP 数据。在第三部分使用我们所生成的数据进行不同模型求解性能测试，并尝试对结果进行分析。第四部分将对本学期工作进行总结，对下学期的工作进行初步规划。

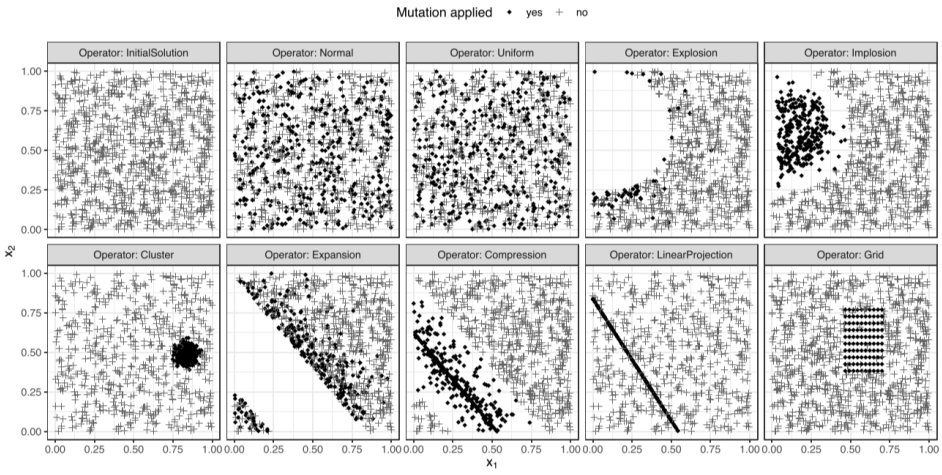
2 数据生成

2.1 论文介绍

论文[9]提到，传统的 TSP 问题生成器主要使用了进化算法，以产生差异较

大的实例。然而，使用这种算法生成的 TSP 实例仍然存在特征与拓扑结构几乎相同的问题。因此，作者在此基础上提出了新的 TSP 实例生成方式，相比传统算法，其生成的 TSP 实例在拓扑结构上更加多样，具有更明显的多种特征，更接近现实世界中 TSP 问题（如大规模集成电路）的实例。

首先生成一个在 0-1 之间均匀分布的点的集合，被称为 RUE(Random Uniform Euclidean)。基于该点集，论文中定义了九种变异算子，这些算子可以通过运算将该集合中的点投影到新的位置。



图表 1 论文中定义的九种 Mutation Operator

当对同一组生成数据迭代多次使用随机算子时，便可以生成一个具有不同随机结构的 TSP 实例。

论文中同时提到了另一种 TSP 实例生成算法，但由于时间有限，我们本次只实现了上文所提到的基于变异算子实现的 TSP 实例生成算法。

2.2 论文复现工作

本阶段中，我们使用 python 复现了论文中的四种变异算子：线性投影(Linear Projection)算子，簇(Cluster)算子，爆破(Explosion)算子，内爆（Implosion）算子。

最终，我们使用由不同算子生成的 TSP 数据训练之前提到的，并对比了不同种类数据集训练出的模型在对应数据集中的表现。

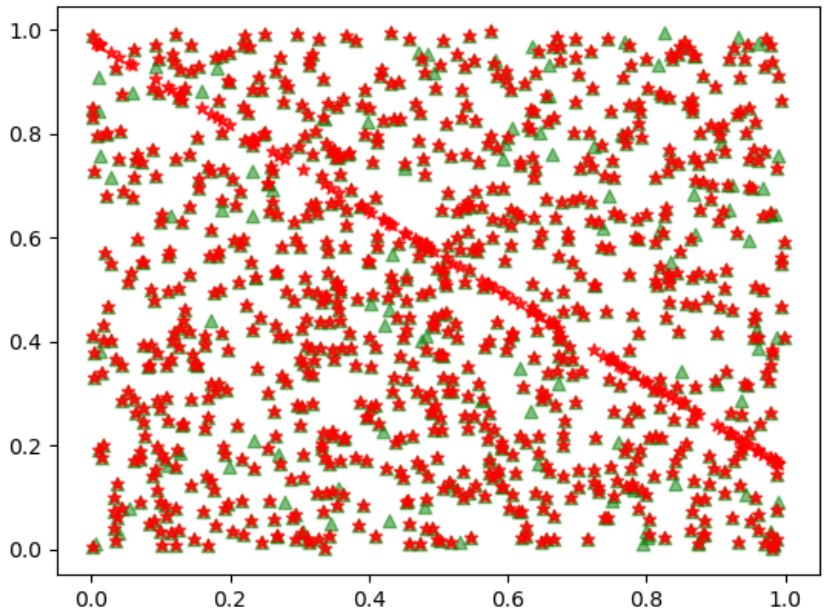
2.3 变异算子

2.3.1 线性投影(Linear Projection)算子

首先按照入参随机选择一定比例的点作为算子的输入点。

生成一个随机的线性方程$f(x) = \beta_0 + \beta_1 \cdot x$，其中$\beta_0$遵循 0-1 之间的均匀分布，为保证该线性方程有大概率落在 0-1 间，若$\beta_0 < 0.5$,则$\beta_1 \sim U[-3,0]$，反之$\beta_1 \sim U[0,3]$

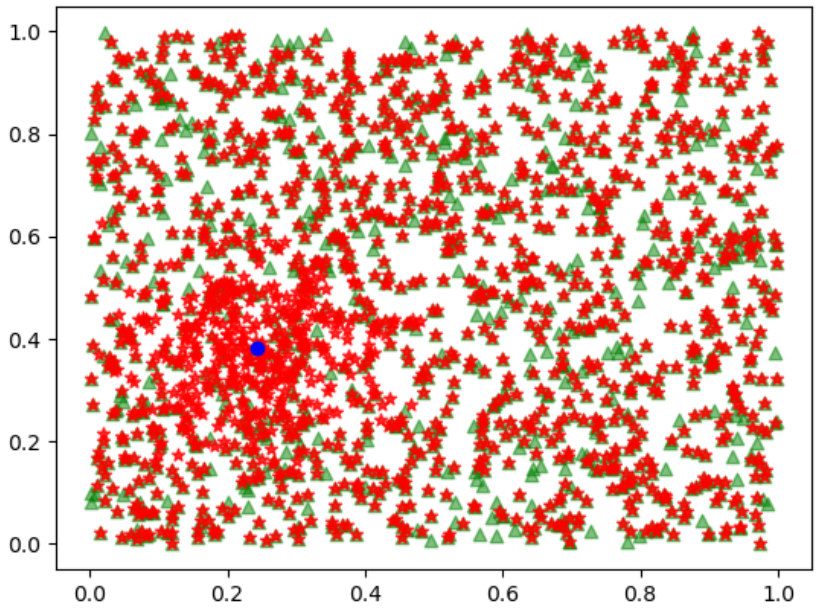
在生成线性方程后，将选定的点的横坐标代入该方程运算，生成新的纵坐标。再在纵坐标上加上正态分布的噪声扰动，为该输入点的投射位置。



图表 2 复现的线性投影算子在 n=1500 的效果

2.3.2 簇(Cluster)算子

首先按照入参随机删除一定比例的点。按均匀分布生成一个随机点$c \in U[0,1]^2$作为中心点。按照分布$N(c, diag(\sigma, \sigma))$生成与删除点同样数量的新点，其中$\sigma \sim U[0.001, 0.3]$



图表 3 复现的簇算子在 n=1500 的效果（蓝色点为中心）

2.3.3 爆破(Explosion)算子

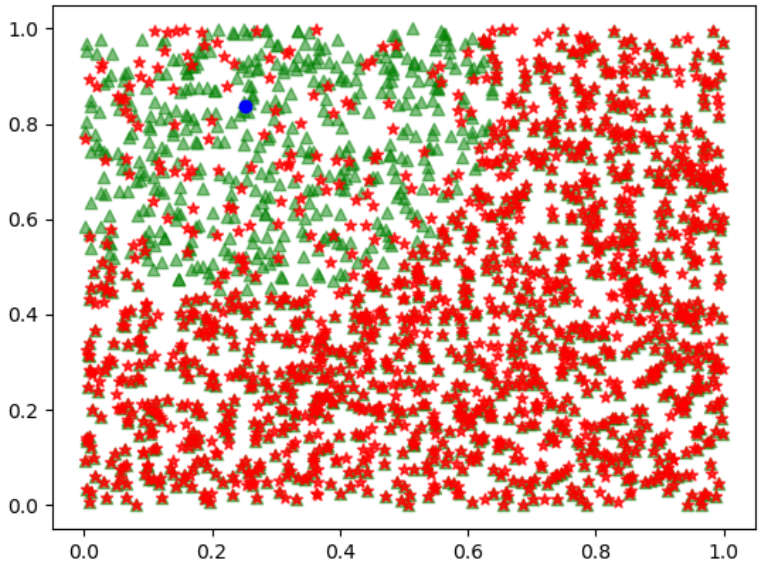
首先按均匀分布生成一个随机点$c \in U[0,1]^2$作为中心点，并根据入参按均匀分布生成一个爆炸半径$r \sim [r_{min}, r_{max}]$。所有以中心点为中心且处于爆炸半径的点均为算子输入点。

按如下公式操作输入点：

$$\pi' = c + (r + s_i) \cdot \frac{c - \pi}{||c - \pi||}$$

其中$s_i \sim Exp(\lambda = \frac{1}{10})$

可能存在点在操作后不在$[0,1]^2$区间中，故在最后将所有越界的点重新按均匀分布确定位置。



图表 4 复现的爆破算子在 n=1500 的效果（蓝色点为中心点）

2.3.4 内爆(Implosion)算子

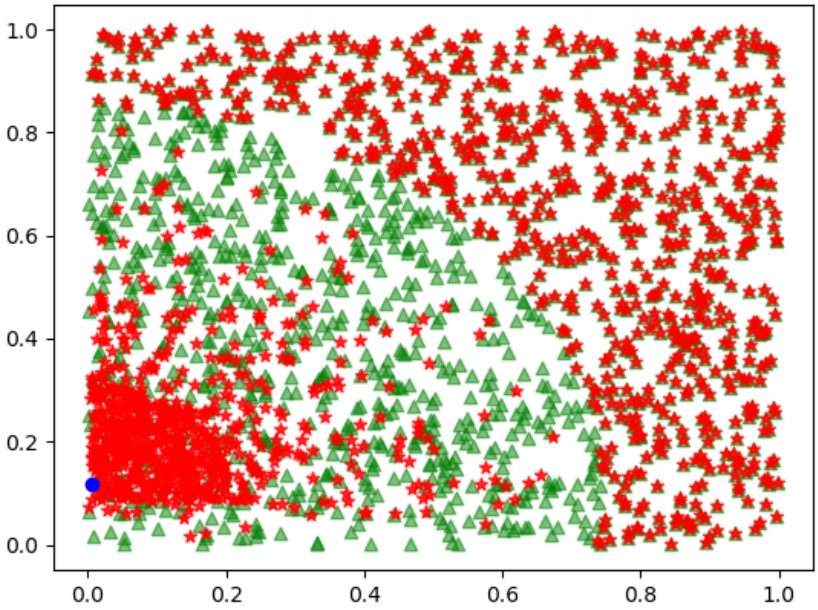
与爆破算子类似，首先按均匀分布生成一个随机点$c \in U[0,1]^2$作为中心点，并根据入参按均匀分布生成一个爆炸半径$r \sim [r_{min}, r_{max}]$。所有以中心点为中心且处于爆炸半径的点均为算子输入点。

按如下公式操作输入点：

$$\pi' = \pi + (c - \pi) \cdot \min(|\tau|, r)$$

其中$\tau \sim N(0,1)$

对于在操作后不在$[0,1]^2$区间中的点，处理方式与爆破算子一致。



图表 5 复现的内爆算子在 n=1500 的效果（蓝色点为中心点）

3 实验

对于使用不同数据集训练出来的 AM, GPN 模型，采用第二阶段使用的过的实验方法测试了它们的求解性能，训练收敛性能，泛化性能和训练效率。在获得了模型性能后，对数据集分布和注意力机制的工作机理进行了简单的探究，尝试对实验结果进行了分析与解释。

3.1 模型性能实验

本实验中使用的设备如表格 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

该实验中对比了使用不同数据集训练的 AM, GCN_NPEC, GPN 模型在不同数据集下的求解性能。

② 实验结果

Problem set	Data size	Mutation	Problem Size
Cluster(1)	320000	Cluster	50
Linear Cluster(2)	320000	L/C combination	50
Linear(3)	320000	Linear	50
Mix(L,C)(4)	320000	Linear and Cluster	50
Uniform(5)	320000	None	50

表格 2 使用数据集介绍

我根据论文[9]，构造了五种数据集用于训练模型。根据表格 2 可以看到，数据集 1，3，4 分别是根据论文[9]中提到的遗传算法，分别使用对应的算子生成。数据集 2 则是由数据集 1，3 组合的来，即两种数据各占一半。

对 AM 模型，我使用了全部五种数据集训练了对应的模型。对于 GPN 模型，由于其性能不及 AM 模型，因此我只使用了数据集 1，2，3，5 对其进行了训练。而对于 GCN_NPEC 模型，由于其未提供开源代码，而在对其复现过程中，发现其极易陷入局部最优导致全局梯度消失。因此参考 AM 模型对其 Decoder 部分进行了修改，并且没有参考论文中的训练指标，改用梯度裁剪 1 和学习率下降更慢的方式进行训练。由于训练数据的问题，依然没有复现论文中其在 VRP 问题上超过 AM 模型的表现，但是经过测试，重新训练的 GCN_NPEC 模型已经可以在某些 VRP 问题上产生比 AM 模型更优秀的解。由于 GCN_NPEC 模型的特殊性，我只测试了其在数据集 5 上的求解性能表现。

	Cluster (1)		Linear Cluster (2)		Linear(3)		Mix (L,C)(4)		Uniform(5)	
	Obj	Gap	Obj	Gap	Obj	Gap	Obj	Gap	Obj	Gap
Concorde	4.705	0%	4.952	0%	5.198	0%	5.203	0%	5.691	0%
AM 50 (1) 100	4.944	5.08%	5.207	5.17%	5.469	5.22%	5.474	5.21%	6.009	5.58%
AM 50 (2) 100	4.911	4.38%	5.143	3.86%	5.384	3.58%	5.384	3.48%	5.901	3.69%
AM 50 (3) 100	4.880	3.72%	5.125	3.50%	5.373	3.38%	5.378	3.37%	5.900	3.68%
AM 50 (4) 100	4.959	5.41%	5.183	4.66%	5.406	4.01%	5.409	3.98%	5.929	4.19%
AM 50 (5) 100	4.827	2.59%	5.063	2.25%	5.299	1.96%	5.304	1.95%	5.795	1.83%

AM 50 (2) 200	4.836	2.78%	5.084	2.67%	5.334	2.61%	5.337	2.57%	5.849	2.79%
AM 50 (2,5) 300	4.809	2.23%	5.054	2.06%	5.295	1.87%	5.301	1.89%	5.800	1.92%
AM 50 (5) 200	4.812	2.29%	5.048	1.94%	5.284	1.66%	5.287	1.61%	5.777	1.51%
GPN 50 (1) 100	4.987	5.99%	5.248	5.98%	5.499	5.80%	5.504	5.80%	6.027	5.90%
GPN 50 (2) 100	5.037	7.06%	5.261	6.23%	5.485	5.51%	5.487	5.46%	6.024	5.86%
GPN 50 (3) 100	5.283	6.87%	5.258	6.19%	5.482	5.47%	5.489	5.49%	6.017	5.73%
GPN 50 (5) 100	5.059	7.51%	5.284	6.71%	5.513	6.06%	5.519	6.06%	6.001	5.61%
GCN_NPEC(5)300									7.315	28.56%

表格 3 使用不同训练数据集训练出来的 AM, GPN, GCN_NPEC 的求解性能表现。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 3 中可以看到，AM 模型在 5 个数据集上的表现都要优于其他竞争对手,绝大多数模型与 Baseline 的差距都小于 5%。使用不同训练集训练的 GPN 与 Baseline 的差距始终要大于 5%，这与第二阶段对于 GPN 模型的认知相似。而 GCN_NPEC 模型，由于其主要用于求解 VRP 问题，因此其在 TSP 问题上表现不佳，但是重新训练的 GCN_NPEC 模型相对于之前还是在 TSP 问题的求解质量上有了长足的进步。

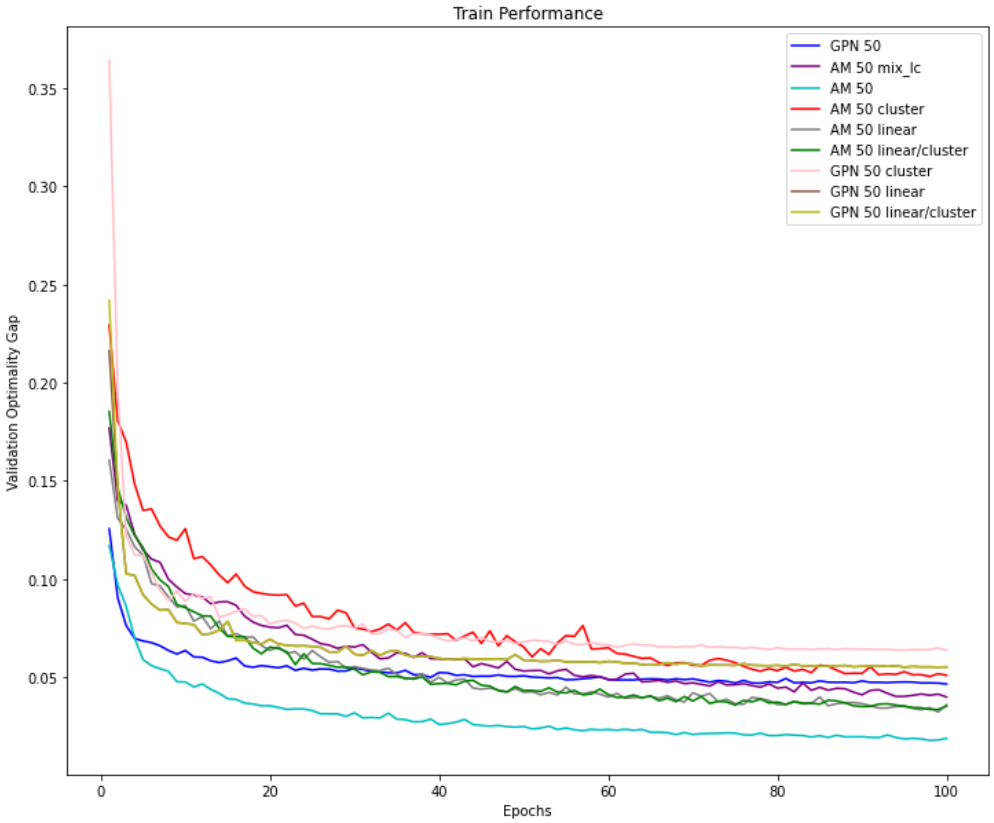
让人出乎意料的是，使用遗传算法生成的带有视觉特征的数据集训练出来的模型，并没有在对应数据集上取得很好的表现。如模型 AM 50 (3) 100(5.373)在(3)数据集上的表现就远不及模型 AM 50 (5) 100(5.299)。而在生成数据集中，含有 Cluster 类型的数据训练出来的模型表现要比不含该类型数据训练出来的模型表现更好。为了进一步探究数据集对模型求解性能的影响，我们分别继续训练了 AM 50 (2) 100 和 AM 50 (5) 100。我们发现，继续使用数据集(2)训练的模型 AM 50 (2) 200，能带来一定的性能提升，但是依然比不过使用数据集(5)训练出来的模型 AM 50 (5) 200。为了探究数据集(5)对模型性能的影响，我们将数据集(2),(5)进行组合，继续训练模型，得到 AM 50 (2,5) 300。通过图表可以看到，其在数据集(1)上取得了 4.809 的成绩，为所有模型的最优解，而在其他数据集上，其与 AM 50 (5) 200 也维持在 0.1%左右。由此我们可以看到，数据集(5)对于模型求解性能提升巨大，而在数据集(5)的基础上拼接其他数据集，可能代来模型求解性能在某一方面的提升。

对于使用不同数据集却没有代来性能提升这一情况，我在 3.2 节中进行了分析，并尝试对其进行了解释。

3.1.2 模型收敛性能测试

① 概述

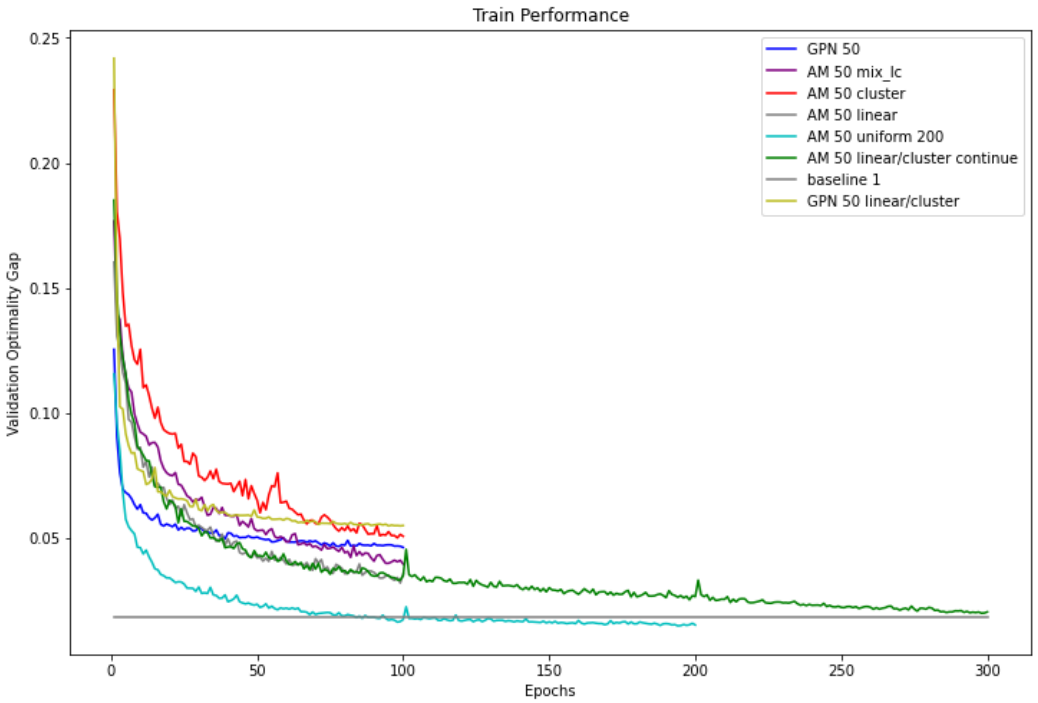
在该实验展现了使用不同数据集训练的 AM,GPN 模型随着训练轮次的增加求解性能的变化。



图表 6 不同数据集下模型的训练效果 图片横轴为训练轮次，纵轴为模型在测试集上的解与 Baseline 的百分比差距

② 实验结果

如图表 6，可以看到使用带有特征数据集(1,2,3,4)训练的模型在训练初期和中期相对于使用数据集(5)训练的模型与 Baseline 的差距减小的更慢，对于 AM 模型和 GPN 模型均是如此。而使用带有特征数据集训练的 AM 模型与使用传统数据的模型相比，表现出了更高的斜率，其模型性能也许有进一步下降的趋势。如 AM 50 Cluster/Linear 在 60-100 轮间的斜率明显高于 AM 50。但对于 GPN 模型，即使换用数据集，其模型的下降趋势始终没有超过使用数据集(5)的 GPN 50。因此，我选择分别对 AM 50 Cluster/Linear 和 AM 50 继续训练。



图表 7 不同数据集下模型的训练效果 图片横轴为训练轮次，纵轴为模型在测试集上的解与 Baseline 的百分比差距(增加了继续训练的模型表现)

如图表 7，可以看到，通过对 AM 50 Cluster/Linear 的继续训练，该模型最终达到了 AM 50 在 100 轮上的求解性能，且其训练斜率仍然存在，而对于 AM 50，虽然继续训练 100 轮让其求解性能进一步提升，但是提升幅度和训练斜率与 AM 50 Cluster/Linear 相比仍较小。由此可以猜想，通过不同数据集的组合，对模型持续训练，可能能让模型的表现达到仅使用数据集(5)所不能达到的高度，但是如何组合数据集以及训练多少轮，仍需要进行更多的实验。

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)1280000	3.85(0.26%)
AM	TSP20(RTX TITAN)	6:53(batch size 512)1280000	5.79(1.66%)
AM	TSP100(RTX TITAN)	24:00(batch size 256)1280000	
GPN	TSP20(RTX TITAN)	4:00(batch size 512)1280000	3.87(0.78%)
GPN	TSP50(RTX TITAN)	24:45(batch size 512)1280000	
GCN_NPEC	CVRP20(RTX TITAN)	4:16(batch size 256)100000	5.04(30.98%)

表格 4 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 4，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。

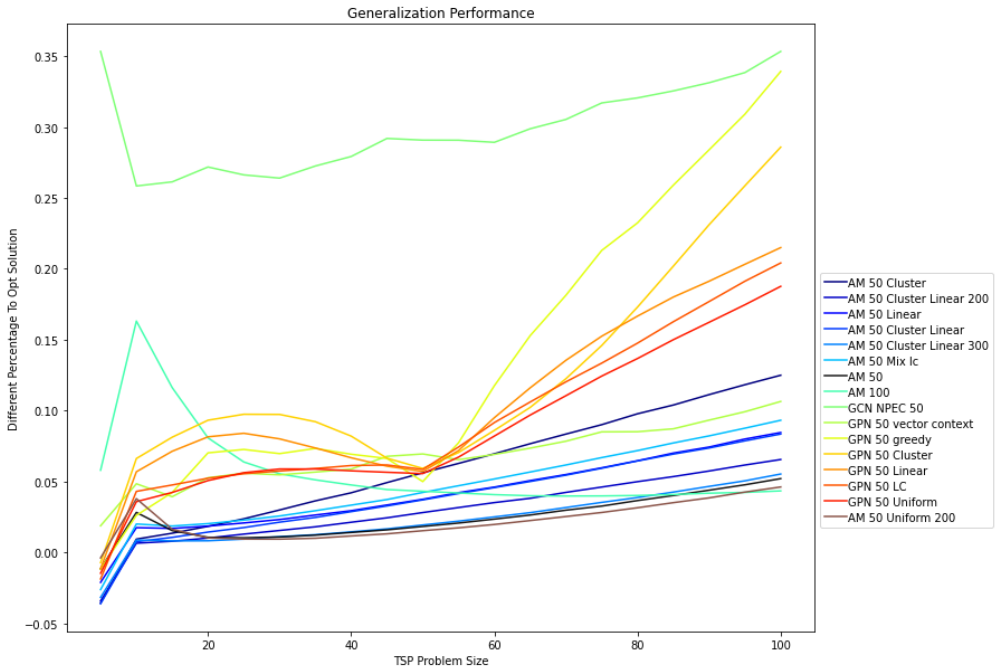
3.1.4 模型泛化性能测试

① 概述

实验对比了不同模型在大小为 Graph size = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 75, 80, 85, 90, 95, 100]的数据集(5)上的泛化性能表现。

② 自身实验

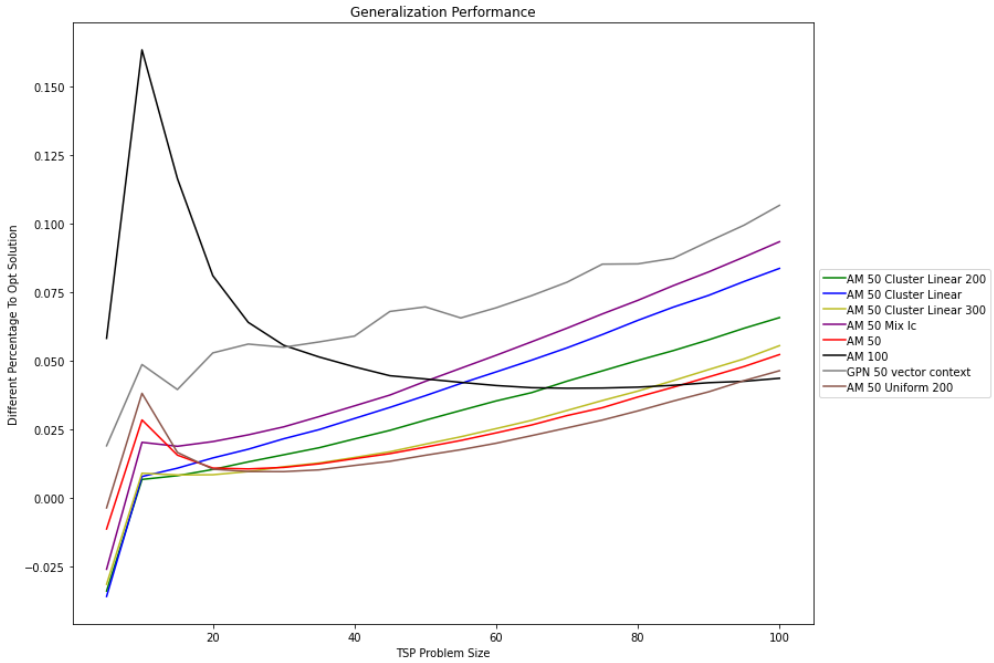
本实验对采用不同数据集训练的模型统一在均匀分布的数据集(5)上做泛化性能测试。首先通过模型自带的数据生成函数生成数据 的对应数据集，其中每个数据集包含个测试样例。将数据转化成 TSPLIB 数据格式使用 Concorde[7]求解器求解出各个数据集的解，每个数据集取平均作为最优解。后分别使用训练好的模型去求解对应数据集，将得到的结果与最优解进行比较，获得泛化性能曲线。



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 模型的泛化性能与其他模型差距较大。其中 GCN_NPEC 模型与 Baseline 的差距始终超过了 25%，而传统 GPN 类的模型则是如果测试集大小偏出训练集大小过多，解的质量迅速下降，与 Baseline 的差距很快超过 10%。但值得一提的是，使用带有特征的训练集训练的 GPN 模型在点较多时有更好的泛化性，而使用了 Vector Context 技术的 GPN 模型泛化性有大幅度提高。

使用不同训练集训练的 AM 模型均表现出了良好的泛化性能，其中泛化性能最好的是 AM 50 Uniform 200。使用带有特征的训练集训练的 AM 模型在问题较小时有更好的泛化性能，但当问题增大时，其泛化性能则不比使用数据集(5)训练的模型，而它们之中与最优模型相差最小的是 AM 50 Cluster Linear 300。



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)

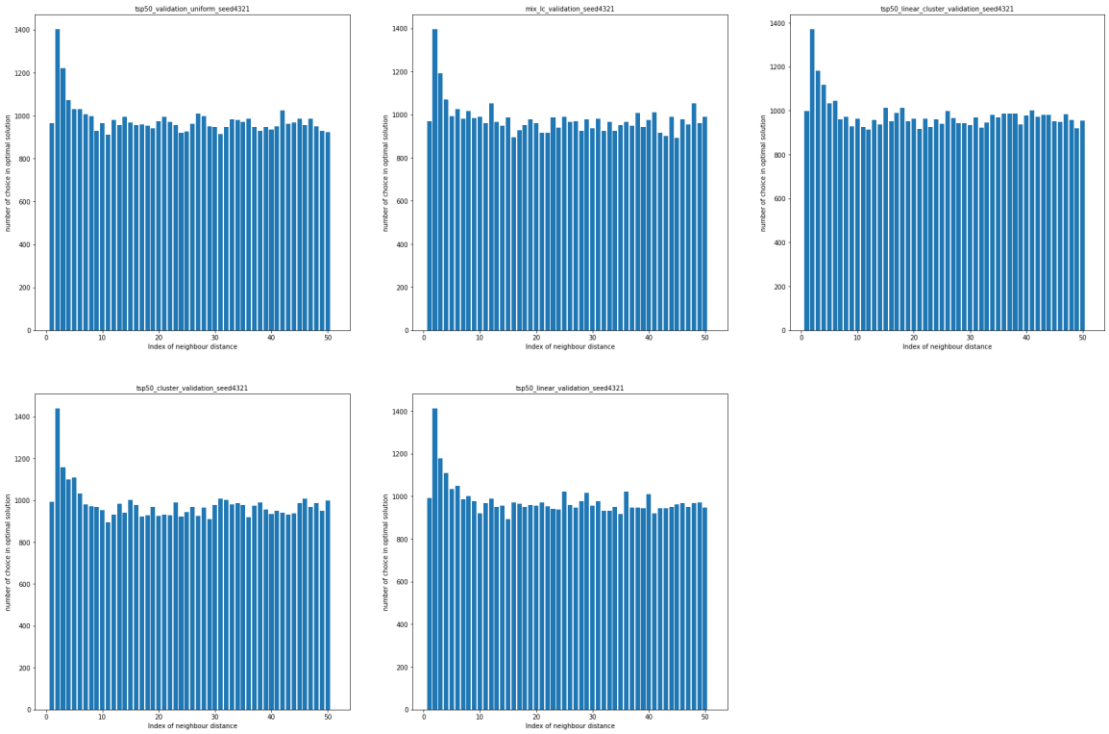
如图表 9，GPN 模型在使用了 Vector Context 后泛化性能有了明显的提升，与 AM 模型的差距大幅度减少。而众多数据集训练出来的 AM 模型泛化性能变化趋势基本相同，有 4 个模型与 Baseline 的差距始终在 5%以内，其中 AM 50 Cluster Linear 200 从未使用均匀分布的数据进行训练，证明了只使用带有特征的训练集训练的 AM 模型也能达到较好的泛化性能。

3.2 额外分析

为了探究使用不同数据集训练出来的模型结果相似这一结论，我使用 Concorde 获得了不同数据集的最优解序列，并统计了最优解在节点选择上的数学性质，发现五个节在节点选择上并没有很大不同。我同时探究了 AM 模型的平均注意力距离随层数和训练数据的变化，并分析了不同模型对输入的位置编码。

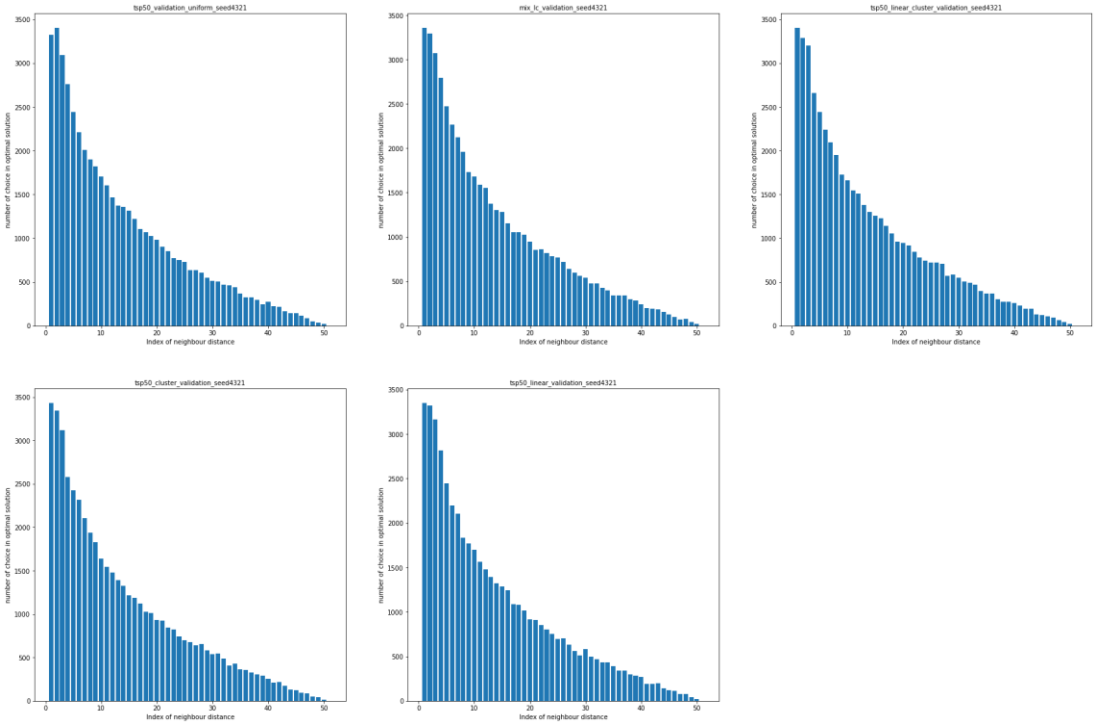
3.2.1 数据集特征分析

我根据 Concorde 获得的最优解序列，统计了最优解序列中下一节点与上一节点的相对距离关系，即下一节点是上一节点第几近的节点。我统计了五个数据集中相对位置关系的出现次数。



图表 10 不同模型节点选择的相对位置关系的统计

如图表 10 所示，在五个数据集的最优解中，下一节点的相对位置关系并没有表现出明显的区别。可以看到在所有数据集中，下一节点是上一节点第二远的节点是出现最多的情况，除此之外，其他相对位置关系出现的次数趋近于相等。但由于不同节点在解序列中出现的顺序不同，在解序列中靠后的点在选择时，可能大多数邻近的点都无法选取，因此我将这些点排除并重新计算相对位置关系，得到图表 11。



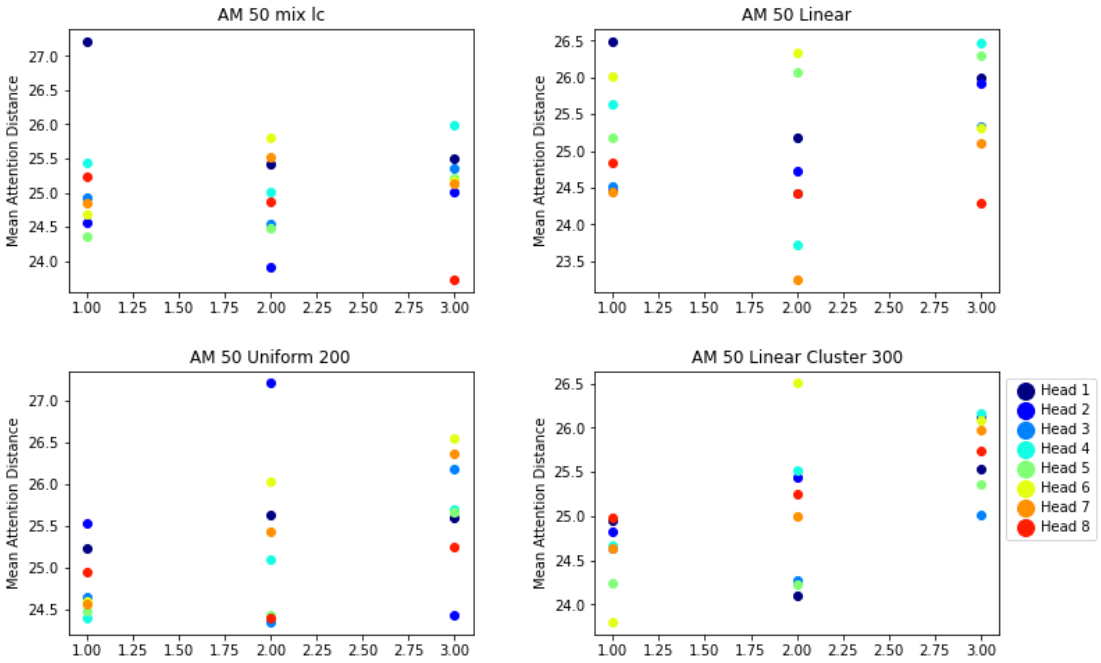
图表 11 不同模型排除掉非法点后节点选择的相对位置关系的统计

如图表 11 所示，在去掉了非法节点后，在五个数据集中，距离当前节点越近的点越有机会成为最优解序列中下一个节点，但遗憾的是，下一节点的相对位置关系并没有表现出明显的区别。

通过上述实验，可以得知，虽然五个数据集视觉上有明显的分布特征，但是在最优解序列中，被选择节点与上一节点的相对位置关系分布并没有体现出明显的不同，推测这可能是不同数据集训练出来的模型在同一测试集下的泛化性曲线并没有显著差距的原因。

3.2.2 注意力距离探究

根据相对位置关系，我测量了 AM 模型多头注意力层所关注的平均注意力距离。该距离是由每个头中每个查询节点与其他所有节点的相对位置通过注意力机制加权得到的数值。在这项测试中，我们使用了数据集(5)的部分数据。



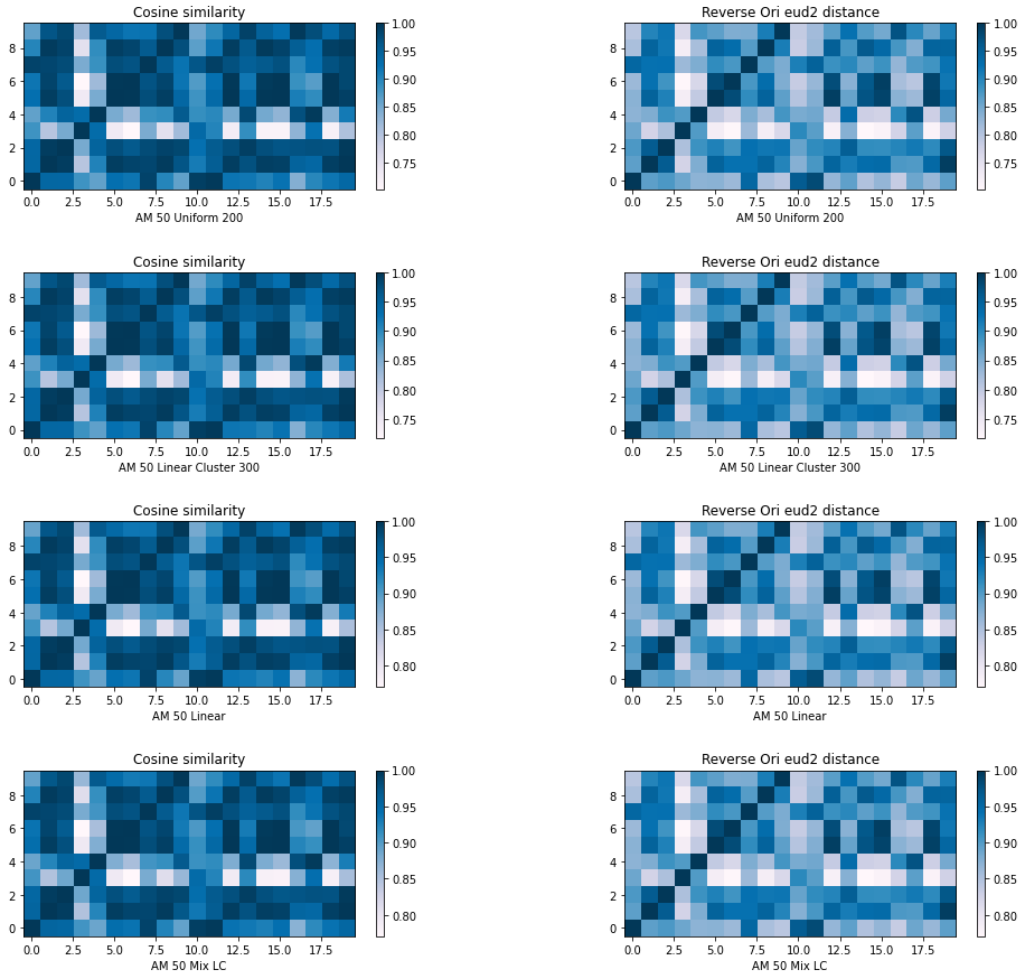
图表 12 不同模型不同层的平均注意力距离，横轴为不同层，纵轴为平均注意力距离

通过图表 12 可以看到，四个模型的平均注意力距离都集中在 23-28 这个区间。其中，除了 AM 50 mix lc 外的其他三个模型，其平均注意力距离都表现出，第 2 层不同头的平均注意力距离差距较大， 而 1，3 层则较小。同时，对于表现较好的 AM 50 Linear Cluster 300 和 AM 50 Uniform 200 两个模型，它们的平均注意力距离随着层数的加深而加深，可以认为随着层数的加深，多头注意力更多的关注较为长距离的关系。

但是，整体来说，不同模型在平均注意力距离上并没有表现出很明显的差距，在随后的实验中我们将会测试不同数据集下不同模型的平均注意力距离会不会发生变化，以及会不会出现明显的差距。

3.2.3 AM 位置编码实验

根据第二阶段实验我们可知，AM 模型对于输入的位置编码与其标准化的欧氏距离之间存在数值相对大小上的相似度，即欧氏距离相对较大的地方，其位置编码之间的余弦相似度也同样较大。这里，我依然采用上一次的实验方式，探究了使用不同数据集训练的 AM 模型的位置编码与其欧氏距离之间的关系。



图表 13 不同模型位置编码的余弦相似度和 EUD2 距离对比

根据图 13，不同模型的位置编码的余弦相似度表现出了很高的相似度，且都与标准化的欧氏距离存在数值相对大小上的相似度。而不同模型位置编码的余弦相似度，在绝对数值方面也表现出了很高的相似度。四个模型的余弦相似度都集中在 0.75-1 这个范围内。

3.3 实验总结

根据一系列的实验，我们探索了采用不同训练数据集的模型的性能表现，并通过对训练数据集，平均注意力距离和位置编码的分析尝试探究了背后的因素。首先，采用不能数据集训练出来的模型在求解性能和泛化性能上并不能产生很大的差距，且绝大多数采用具有明显特征的训练数据的模型表现都逊色于采用均匀采样的 AM 模型。但是，通过将多种训练数据进行混合，我们也可以训练出与最优 AM 模型相近的模型，且在一些数据集上表现优于最优 AM 模型。而在原因探索上，不同 AM 模型在位置编码上并没有表现出明显的区别，而在平均注意力距离上虽然有所不同，但是在绝对数值上差距依然不大。而在对数据集的分析上，我揭示了虽然不同数据集在视觉上具有明显特征，但是在被选择节点的相对位置关系上，并没有产生很大的差距，故而推测这是不同模型表现差距不大的原因。至于不同数据集对于模型训练的效果，以及平均注意力距离的更多表现，仍然等待着我们进一步探索发掘。

4 总结与规划

4.1 项目总结

在本学期的项目中，我们对旅行商问题的神经网络求解器进行了深入的探索。我们从复现该领域经典模型出发，不断从优秀论文中汲取测试方法，最终形成一套针对神经网络求解器泛化和求解性能的完备测试。随后，我们将目标转向训练数据，通过参考论文[9]，我们生成了带有明显视觉特征的 TSP 数据集，并使用它们分别训练了该领域的经典模型，并将我们之前总结出来的测试方法应用在它们身上，测出它们的泛化性能和求解性能。而面对不符合直觉的结果，我们参考相关论文，设计实验对数据集，平均注意力距离和位置编码进行分析，尝试对产生的结果进行了解释。

4.2 未来规划

通过这学期的实验，我们对神经网络求解器有了基本的了解，而在旅行商问题这一领域，我们积累的经验，对领域经典模型和它们的性能有了大概的了解。现阶段，对于如何生成并组合训练数据，从而试图提高模型的泛化和求解性能，仍然等待着我们的进一步探索与发现。同时，有关平均注意力距离和位置编码所

展现出来的模型特质，也等待着我们的进一步探索。增加层的数量是否能在平均注意力距离实验上有所体现？位置编码是否有进一步优化的空间，以便提高解的质量，或者是将问题拓展到 VPR 等约束更强的问题上去？这些疑问都等待着我们去解决。

5 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/*

[9] Bossek, J., Kerschke, P., Neumann, A., Wagner, M., Neumann, F., & Trautmann, H. **Evolving diverse TSP instances by means of novel and creative mutation operators.** In: *Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, 2017.*