arXiv:2305.01883v2 [cs.LG] 6 Mar 2024

一种用于学习旅行商问题的轻量级 CNN-Transformer 模型

Minseop Jung[1], Jaeseung Lee[1], Jibum Kim[1*]

[1] 仁川国立大学计算机科学与工程系，仁川，22012，韩国。

*通讯作者。电子邮箱：jibumkim@inu.ac.kr；贡献作者：minseob22@inu.ac.kr；hunni10@inu.ac.kr；

摘要

多项研究尝试使用各种深度学习技术解决旅行商问题（TSP）。其中，基于 Transformer 的模型即使在大规模旅行商问题（TSP）上也展现出最先进水平的性能。然而，这些模型基于全连接注意力模型，面临较高的计算复杂度和 GPU 内存使用量。我们的工作是首个基于 CNN嵌入层和部分自注意力、用于解决 TSP 的 CNN–Transformer 模型。与标准基于 Transformer 的模型相比，我们的 CNN-Transformer 模型能够利用 CNN嵌入层从输入数据中更好地学习空间特征。它还通过所提出的部分自注意力，消除了全连接注意力模型中的大量冗余。实验结果表明，所提出的 CNN嵌入层和部分自注意力在提升性能和计算复杂度方面非常有效。所提出的模型在真实世界数据集上展现出最佳性能，并在多个方面优于其他现有的最先进水平（SOTA）基于 Transformer 的模型。我们的代码公开发布于https://github.com/cm8908/CNN Transformer3。

关键词：旅行商问题，组合优化问题，CNN-Transformer，轻量级模型

1 引言

旅行商问题（TSP）是计算机科学和运筹学中的经典 NP 困难问题，旨在寻找一条尽可能短的路线，以恰好访问每个

城市一次并返回起点城市 [1]。当城市数量较多时，寻找 TSP 的最优解在计算上代价高昂。研究人员研究了各种启发式方法和近似算法，这些方法和算法能够在合理的时间内为该问题提供高质量的解。

最简单且最流行的启发式方法之一是最近邻启发式。它从随机选取的城市出发，只要还有未访问的城市，就反复选择距离最近的未访问城市作为下一个要访问的城市。最后，它返回起始城市，完成巡回。另一个著名的 TSP 启发式算法是 Christofides 算法 [2]。它利用表示这些城市的图的最小生成树来寻找近似解；这棵树连接了所有城市，且总边权尽可能小。该算法已知能够保证找到的解与最优解的差距不超过 3/2 倍。另一种使用启发式方法求解 TSP 的著名工具是 Google-OR 工具 [3]。它通过局部搜索和元启发式方法，为包括 TSP 和车辆路径问题在内的多种组合优化问题寻找近似解。然而，启发式方法以计算成本换取最优性，并以规则的形式表示 [4, 5]。

对于许多旅行商问题实例，Concorde 被认为是速度最快、最精确的旅行商问题求解器，能够求得最优解 [6]。Concorde 使用带割平面的整数规划求解器和分支定界法。它假设旅行商问题是对称的，即两个城市之间的距离在两个相反方向上相同 [7]。Gurobi 也能求得旅行商问题的最优结果，但由于 Concorde 专门针对旅行商问题，因此速度比 Gurobi 更快 [4, 8]。

许多研究基于深度学习探索旅行商问题的近似解法。其中，开创性的工作是指针网络[7]。这是一种基于监督学习的方法，使用基于循环神经网络的编码器和解码器。在实验中，研究假设旅行商问题是平面上的对称问题，并采用集束搜索解码过程来排除无效巡回路线，例如重复访问同一城市或遗漏目的地。Bello 等人采用基于强化学习的方法更新长短期记忆网络模型的学习参数，并将巡回路线长度作为负奖励信号 [9]。Nazari 等人使用嵌入代替指针网络中的循环神经网络编码器，在不影响性能的前提下降低了计算复杂度 [10]。Joshi 等人提出了一种方法，通过图卷积神经网络模型和基于监督学习的方法预测整个图的边概率矩阵 [11]。Stohy 等人提出了一种用于旅行商问题的混合指针网络模型，在大规模旅行商问题实例上表现良好 [12]。然而，与基线图指针网络相比，该模型结构更复杂，因此推理时间较长 [13]。研究者还进行了多项尝试，将卷积神经网络（CNN）引入旅行商问题。研究者将二维卷积用于旅行商问题，但未能取得良好性能[14, 15]。Sultana 等人提出了一种结合一维卷积神经网络与长短期记忆网络的新模型，但它仍然是基于循环神经网络的模型 [16]。

近期，基于注意力机制的 Transformer 模型在多个研究领域展现出良好的性能 [17–20]。研究人员成功使用基于 Transformer 的模型为旅行商问题寻找近似解 [4, 5, 21–25]。Deudon等人提出了一种新颖的

深度强化学习方法，用于求解旅行商问题。模型以城市坐标作为输入，并通过强化学习进行训练，以预测城市序列的概率分布 [21]。Kool 等人提出了一种完全由注意力块构成的基于 Transformer 的模型，并使用 REINFORCE 训练该模型，以求解旅行商问题和车辆路径问题等多种路径规划问题 [4]。Wu 等人提出了一种基于 Transformer 的深度强化学习框架，该框架训练一种改进启发式算法，通过迭代不断改进初始解 [22]。研究人员提出了一种方法，将多次滚动展开和数据增强应用于 Kool 的注意力模型 [23]。

最近，Bresson 等人提出了一种 TSP Transformer 模型 [5]。该模型基于采用多头注意力和残差连接的标准 Transformer 编码器，但使用批归一化代替层归一化。该模型采用自回归解码方法，并在解码器部分引入了一个自注意力块。它通过自注意力模块，利用部分路径中的所有城市构建查询[5]。他们展示了该模型在各种旅行商问题实例上达到了最先进水平（SOTA），并报告称，在 TSP50 和 TSP100 上，其最优性差距分别为 0.0004% 和 0.39%。尽管 TSP Transformer 模型在许多旅行商问题实例上表现出最先进的性能，但其模型结构较为复杂，基于全连接注意力模型。此外，该模型还需要大量的 GPU 使用量。而且，其训练和推理时间都非常长 [26]。最近，已有多项研究致力于降低标准 Transformer 模型的计算复杂度 [27–29]。针对旅行商问题，近期有一项研究通过移除可学习解码器，使模型更加轻量化 [24]。另一项针对轻量级 TSP Transformer 模型的类似研究也已开展[25]。Yang 等人提出了一种用于旅行商问题的内存高效的基于 Transformer 的模型，称为 Tspformer。与标准基于 Transformer 的模型相比，该模型成功降低了 GPU/CPU 内存使用量 [25]，但其解的质量不如 SOTA 模型。

Pan 等人提出了一种基于分层强化学习的构造式方法（H-TSP），专门用于求解大规模旅行商问题实例 [30]。该方法采用分层深度强化学习，在上层和下层分别设置策略。尽管 H-TSP 在求解大规模旅行商问题实例方面表现出色，但它需要通过预训练为低层模型额外设置一个预热阶段，且训练过程相当复杂。因此，该方法存在计算资源需求量大、训练时间长的缺点。Ren 等人采用自监督强化学习方法求解动态旅行商问题 [31]。该论文提出了一种结合自注意力和上下文注意力机制的新特征提取机制 [31]。该方法的优势在于无需人工设计奖励函数。

本文提出了一种新颖的 CNN-Transformer 模型，通过仅对解码器中最近访问过的节点执行注意力，来实现部分自注意力。标准 Transformer模型中的线性嵌入未考虑局部空间信息，在学习局部组合性方面存在局限。因此，我们在标准 Transformer模型中加入 CNN嵌入层，以提取输入数据的局部空间特征，因为 CNN 擅长学习欧几里得空间中节点的空间不变性。其次，标准 Transformer模型是

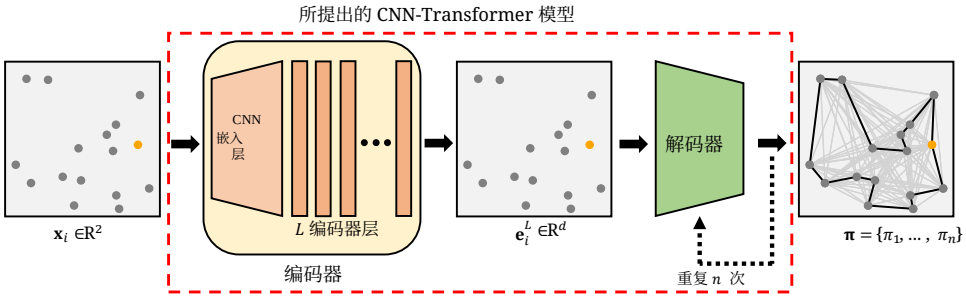


图 1 所提出的用于旅行商问题的 CNN-Transformer 模型概览

基于全连接注意力模型 [26]。因此，它面临着巨大的计算复杂度和内存消耗。此外，由于 Transformer 模型采用全连接拓扑结构，其在学习局部组合性方面存在不足。针对旅行商问题（TSP），我们提出了一种部分自注意力机制，仅关注解码器中最近访问过的节点，从而改进注意力机制。我们的研究发现，在解决 TSP 时，Transformer 模型的全连接拓扑中存在大量冗余。通过移除过多的注意力连接，这种冗余的减少提高了旅行商问题解的质量。本文的主要贡献总结如下：

- 据我们所知，我们提出了首个用于学习 TSP 解的 CNN-Transformer 模型。我们的结果表明，CNN 嵌入层在学习各种旅行商问题实例的局部空间特征方面非常有效。
- 所提出的模型基于部分自注意力机制，该机制仅对解码器中最近访问过的节点执行注意力计算。因此，与基于全连接拓扑的标准Transformer模型相比，所提出的模型能够更好地学习局部组合性。
- 通过移除解码器中冗余的注意力连接，所提出的模型显著降低了 GPU 内存使用量，与标准 Transformer模型相比，推理时间也大幅缩短。

2 提出的模型

我们提出了一种 CNN-Transformer 模型，它将卷积神经网络嵌入层与标准 Transformer模型相结合。所提出的 CNN-Transformer 模型采用编码器和解码器架构。编码器中包含一个 CNN嵌入层，用于提取局部空间信息。我们基于部分自注意力改进了注意力机制，从而去除了不必要的注意力连接。

图1 展示了所提出模型的整体结构。所提出模型的输入是一个包含 n 个节点（城市）的平面点集 $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$，其中$\mathbf{x}_i \in \mathbb{R}^2$ 表示这些点的二维笛卡尔坐标。该模型的输出是七码

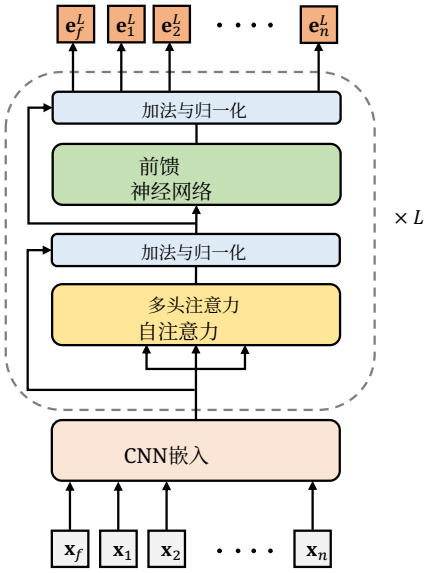


图2 采用CNN嵌入层和 L 个相同编码器层的编码器架构

模型，记为 $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_n\}$，是一个表示预测最优巡回路线的序列，其中 π_t 是解码器在第 t^{th} 个解码步骤中选取的节点索引。设 $D(\mathbf{x}_i, \mathbf{x}j)$ 为节点 $\mathbf{x}_i$ 和 $\mathbf{x}j$ 之间的距离。我们的目标是在恰好访问每个节点一次后返回起始节点，从而最小化总巡回路线长度 $\sum_{t=1}^{n-1} D(\mathbf{x}_{\pi_t}, \mathbf{x}_{\pi_{t+1}}) + D(\mathbf{x}_{\pi_n}, \mathbf{x}_{\pi_1})$。

2.1 编码器

所提出的编码器由CNN嵌入层和 L 个相同的编码器层组成，如图2所示。CNN嵌入层通过从输入数据点中提取局部空间信息来生成嵌入向量，并将其传递给后续编码器层。

每个编码器层由两个子层组成：多头自注意力（MHSA）子层和逐点前馈（FF）子层。MHSA子层执行多头自注意力，以捕获各节点之间的依赖关系；逐点FF执行非线性激活。在每个子层之间加入了残差连接 [32] 和批归一化 [33] 。与先前的研究 [4, 5], 类似，我们使用批归一化而非层归一化，因为它能够有效处理大量节点。

2.1.1 CNN嵌入层

我们在编码器中加入CNN嵌入层，以从输入节点中提取空间信息。令 $\mathbf{X}_i^{k\text{-NN}}$ 为 i^{th} 节点的特征向量与距离 i^{th} 节点最近的 k 个邻近节点特征向量的拼接结果。然后，编码器中 i^{th} 节点的嵌入向量 $\mathbf{x}_i^{\text{emb}}$ 的计算方式为

作为 i^{th} 输入节点的节点嵌入之和，$\mathbf{x}_i\mathbf{W}_{\text{emb}}$，以及 $\mathbf{X}_i^{k\text{-NN}}$ 的卷积，表示为：

$$\mathbf{x}_i^{\text{emb}} = \mathbf{x}_i\mathbf{W}_{\text{emb}} + \text{Conv}(\mathbf{X}_i^{k\text{-NN}}) \in \mathbb{R}^d, \tag{1}$$

其中$\mathbf{W}_{\text{emb}} \in \mathbb{R}^{2\times d}$ 是用于节点嵌入的可学习参数。采用固定的卷积核大小 k+1 ，以确保 Conv($\mathbf{X}_i^{k\text{-NN}}$)与节点嵌入$\mathbf{x}_i\mathbf{W}_{\text{emb}}$处于相同的 d维空间中。

2.1.2 编码器层

编码器包含 L 个相同的编码器层，第一个编码器层以 CNN嵌入层的输出$\mathbf{x}_i^{\text{emb}}$作为输入。每个编码器层包含两个子层：多头自注意力子层和逐点前馈子层。

多头自注意力子层。 l^{th} 编码器层的多头自注意力子层输入$\mathbf{E}^{l-1}$是第$(l-1)^{\text{th}}$ 个编码器层的输出。 l^{th} 编码器层的多头自注意力子层输出$\hat{}\mathbf{E}^l$，通过先对$\mathbf{E}^{l-1}$应用多头自注意力（MH^l），再进行残差连接和批归一化（BN^l）得到。函数 MH^l(Q, K, V) 接收三个输入 Q、K、V，分别表示查询、键和值向量，用于在第 l^{th} 个编码器层执行多头注意力机制。多头自注意力子层的输出$\hat{}\mathbf{E}^l$表示如下：

$$\hat{\mathbf{E}}^l = \text{BN}^l\left(\mathbf{E}^{l-1} + \text{MH}^l\left(\mathbf{E}^{l-1}, \mathbf{E}^{l-1}, \mathbf{E}^{l-1}\right)\right), \tag{2}$$

其中 $\mathbf{E}^0 = \{\mathbf{x}f、\mathbf{x}_1^{\text{emb}}、\text{.....}、\mathbf{x}_n^{\text{emb}}\} \in \mathbb{R}^{(n+1)\times d}$。这里，$\mathbf{E}^0$, 是第一个编码器层的输入，由起始标记 $\mathbf{x}f$ 与 $\{\mathbf{x}_1^{\text{emb}}、\text{.....}、\mathbf{x}_n^{\text{emb}}\}$拼接而成。我们将 $\mathbf{x}f$加入其中，构成一个虚拟节点特征向量，以学习与其他节点特征之间的依赖关系，从而使解码能够从最佳可能位置 [5]开始。

逐点前馈子层。 l^{th} 编码器层中的逐点前馈子层由两个线性投影和 ReLU激活构成，其输入为 $\hat{}\mathbf{E}^l$。该子层先进行非线性激活，随后执行残差连接和批归一化，并生成输出 $\mathbf{E}^l$，记作：

$$\mathbf{E}^l = \text{BN}^l\left(\hat{\mathbf{E}}^l + \text{FF}^l\left(\hat{\mathbf{E}}^l\right)\right), \tag{3}$$

其中，FF^l 是 l^{th} 编码器层的前馈子层。

编码器输出。 令$\mathbf{e}_i^L$ 分别表示 i^{th} 编码器层中 L^{th} 节点的编码器输出， f 表示起始标记的索引。最后，生成 L^{th}编码器层的编码器输出$\mathbf{E}^L = \{\mathbf{e}_f^L，\ \mathbf{e}_1^L，\ \text{.....}，\ \mathbf{e}_n^L\}$，并将其输入解码器。

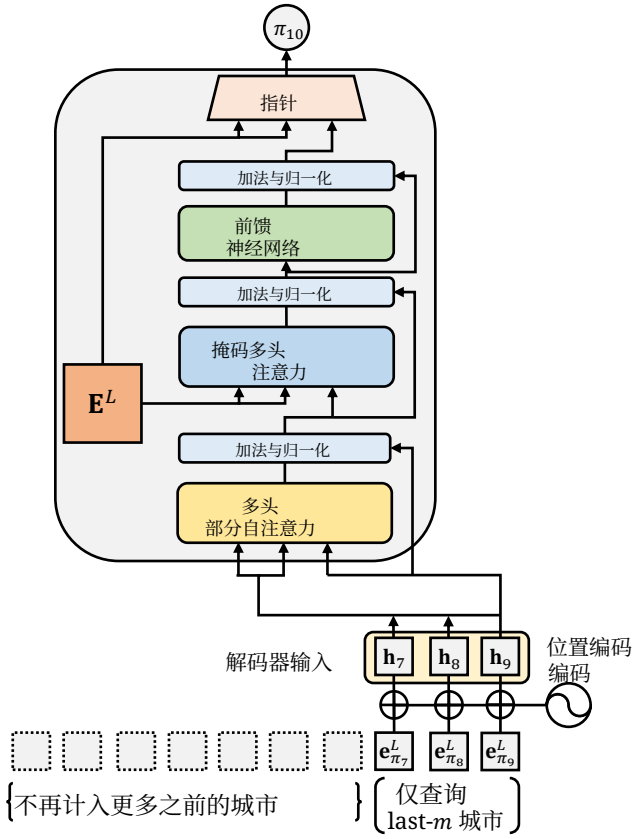


图 3 所提出的解码器架构。该图展示了当前时间步为 $t = 10$ 且参考向量数量为 $m = 3$ 时的解码过程

2.2 解码器

<escape>我们采用自回归方式进行解码，每次处理一个节点。解码器由四层组成，每层之后均接有残差连接和层归一化。图3 展示了一个解码过程示例，其中当前时间步（t）为 10，参考向量的数量（m）为 3。</escape>

第一层是多头部分自注意力层（MHPSA），通过对先前步骤中已访问节点的编码器输出执行注意力，提取过去的信息。与以往的研究不同，我们使用更少的参考向量，以提升性能和计算效率。第二层是掩码多头注意力层（MMHA），它执行一种注意力机制，其中查询是 MHPSA 层的输出，参考向量则是未访问节点的编码器输出。第三层是逐点前馈层，执行线性投影和非线性激活，类似于

编码器中的逐点前馈子层。指针层通过计算未访问节点上的概率分布，选择下一个要访问的节点。

MHPSA层。我们的研究动机在于，与较早访问的节点相比，最近访问的节点与当前步骤中待选择的节点更加相关。基于这一直观事实，所提出的部分自注意力仅对最近访问的节点执行注意力计算。我们期望，与以往基于全连接注意力的方法相比，所提出的模型能够更好地学习局部组合性。

MHPSA使用当前时间步 t的解码器输入$\mathbf{h}_t$作为查询，并使用最近访问节点的解码器输入作为参考向量来执行自注意力。其计算方式如下：

$$\mathbf{h}_t = \mathbf{e}^L_{\pi_{t-1}} + \mathrm{PE}_t, \tag{4}$$

其中，$\pi_0 = f$ 和PE_t 表示时间步 t的位置编码。

所提出的部分自注意力仅使用最近访问的 m 个节点的解码器输入作为参考向量。例如，假设当前时间步为 t，则将时间步 $t-m$ 至 t 的解码器输入作为参考向量，记为$\mathbf{H}_t = \{\mathbf{h}_{t-m}, \ldots\ldots, \mathbf{h}_t\} \in \mathbb{R}^{m\times d}$。因此，内存使用量和计算时间^均显著降低。MHPSA层的输出$\mathbf{h}_t$计算如下：

$$\hat{\mathbf{h}}_t = \mathrm{LN}\left(\mathbf{h}_t + \mathrm{MH}^{L+1}\left(\mathbf{h}_t, \mathbf{H}_t, \mathbf{H}_t\right)\right), \tag{5}$$

其中 LN 表示层归一化。

MMHA 层。掩码多头注意力层使用 $\hat{\mathbf{h}}_t$ 作为查询，并使用$\mathbf{E}^L$ 作为参考向量。我们采用与 [5] 相同的掩码多头注意力层。令 $\zeta_t \in \mathbb{R}^{n+1}$ 为掩码，其中，对于注意力权重，未访问节点对应的值为 1，已访问节点对应的值为 0。然后，MMHA 层的输出计算如下：

$$\tilde{\mathbf{h}}_t = \mathrm{LN}\left(\hat{\mathbf{h}}_t + \mathrm{MMH}^{L+1}\left(\hat{\mathbf{h}}_t, \mathbf{E}^L, \mathbf{E}^L, \zeta_t\right)\right), \tag{6}$$

其中，MMH(Q, K, V, ζ) 是 MH(Q, K, V) 的修改函数，它额外接收一个输入掩码 ζ，并将注意力函数 [17] 替换为掩码注意力函数，定义如下：

$$\mathrm{MaskedAttention}(Q, K, V, \zeta) = \mathrm{softmax}\left(\frac{QK^\intercal}{\sqrt{d_k}} \odot \zeta\right) V,$$

其中，$\odot$ 表示逐元素乘积运算。

˜逐点前馈层。逐点前馈层的输入为 $\mathbf{h}_t$，输出为¯$\mathbf{h}_t$，记作：

$$\bar{\mathbf{h}}_t = \mathrm{LN}\left(\tilde{\mathbf{h}}_t + \mathrm{FF}^{L+1}\left(\tilde{\mathbf{h}}_t\right)\right). \tag{7}$$

指针层。 指针层的目标是计算未访问节点上的概率分布。我们使用 $\bar{\mathbf{h}}_t$ 作为查询，并使用 $\mathbf{E}^L$ 作为参考向量执行单头注意力。我们将注意力权重用作概率分布 p_t，以确定下一个要访问的节点。使用掩码避免访问已访问的节点。然后， p_t 可计算如下：

$$p_t = \text{softmax}\left(c \cdot \tanh\left(\frac{qK^\intercal}{\sqrt{d}}\right) \odot \zeta_t\right), \tag{8}$$

其中 q 和 K 分别是单头注意力中的查询向量和参考向量，且 c 是一个用于控制逻辑值范围的超参数 [9]。

在训练阶段，解码器将 p_t 视为分类分布，以便对节点索引进行采样；在推理阶段，则选择概率最高的节点索引。重复此过程 n 次，得到节点索引序列 $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_n\}$，即模型的最终输出。

2.3 基于强化学习的模型训练

我们通过强化学习训练模型，并将巡回路线长度作为负奖励。损失函数为平均路径长度。设 θ 为模型训练参数。那么， $p(\boldsymbol{\pi};\theta)$表示模型生成 $\boldsymbol{\pi}$的概率，其定义如下：

$$p(\boldsymbol{\pi};\theta) = \prod_{t=1}^{n} p(\pi_t|\pi_1, \ldots, \pi_{t-1};\theta), \tag{9}$$

其中， $p(\pi_t|\pi_1, \ldots, \pi_{t-1};\theta)$表示在时间步 t从 p_t 中选择 π_t 的概率。我们使用REINFORCE算法更新 θ。将 θ的一个副本 θ_b用作基线。我们将模型参数为 θ_b 时以贪心方式生成的索引序列记为 $\boldsymbol{\pi}'$ 。设 $\ell(\boldsymbol{\pi})$和 $\ell(\boldsymbol{\pi}')$分别为训练模型和基线模型所生成节点序列的总巡回路线长度。然后，使用REINFORCE算法，通过梯度下降方法优化REINFORCE损失 $L(\theta)$：

$$L(\theta) = \mathbb{E}_{\boldsymbol{\pi}\sim p(\boldsymbol{\pi};\theta)}\left[\ell\left(\boldsymbol{\pi}\right) - \ell\left(\boldsymbol{\pi}'\right)\right]. \tag{10}$$

$L(\theta)$ 的梯度计算如下：

$$\nabla_\theta L(\theta) \approx \sum_{\boldsymbol{\pi}} \left(\ell\left(\boldsymbol{\pi}\right) - \ell\left(\boldsymbol{\pi}'\right)\right) \nabla_\theta \log p(\boldsymbol{\pi};\theta). \tag{11}$$

我们在一个训练轮次中使用 $\nabla_\theta L(\theta)$ 优化训练模型。一个训练轮次结束后，我们比较训练模型和基线模型的平均路径长度。如果训练模型的平均路径长度短于基线模型，我们将θ 复制到 θ_b 。

3 实验

3.1 数据集

随机数据集。 我们假设一个二维平面对称旅行商问题，其中两个城市之间相反方向的距离相等。在模型训练和验证过程中，我们使用均匀分布在线生成随机数据，数据规模为$[0, 1] \times [0, 1]$。我们生成了 10,000 个测试实例。我们在从 $n = 50$ （TSP50）到 200 个节点（TSP200）的旅行商问题上进行训练和测试。使用 Concorde[6]生成的巡回路线被视为精确解和测试实例的标签。

TSPLIB 数据集。 TSPLIB [34]是一个广泛使用的基准数据集，用于评估旅行商问题求解器在具有不同分布的各种真实世界数据上的性能。我们从 TSPLIB 中选取了十个被认为相对困难的二维欧几里得问题实例。设 N 和 A 分别为节点数和节点覆盖的正方形面积 [16]。然后，我们使用临界参数值 $\frac{l}{\sqrt{N \cdot A}}$来评估 TSPLIB 的难度，其中 ~~$l$ 为~~最优巡回长度 [35, 36]。临界参数值接近 0.75 表示难度更高。我们对每个 TSPLIB 实例进行归一化，使所有 TSPLIB 实例均处于 $[0, 1] \times [0, 1]$。

3.2 超参数与解码策略

超参数。 我们没有对所提出模型的超参数进行调优。对于所有 TSP问题规模，我们都使用相同的超参数。所提出的模型包含六个编码器层（$L = 6$），且采用 $k = 10$。CNN 嵌入层的卷积核大小设为 11。模型的隐藏维度为 128（$d = 128$），每个逐点前馈层（也称子层）的隐藏维度设为 512。模型包含八个注意力头。解码器中的 logit 范围裁剪值 c 设为 10。

在模型训练中，我们使用 Adam 优化器，学习率固定为 0.0001。批大小设为 512。我们使用训练数据训练 100 轮，但进一步增加训练轮数可能会提升模型性能。我们的实验在一台 AMD EPYC 7513 32 核处理器和一块 Nvidia A6000 GPU 上进行。

解码策略。 在测试时，我们采用贪心算法和束搜索进行解码。束搜索 [37]是一种广度优先搜索策略，在每个解码时间步考虑排名前 B 的候选项，并在解码结束时选择最佳解。我们将束宽（B）设为 2,500，以便将我们的结果与其他采用 2,500 束宽的最先进的模型 [5]进行比较。

3.3 实验列表

在我们的研究中，我们进行了四项实验，以评估所提出模型的性能。第三项实验同时使用了随机数据集和 TSPLIB 数据集，而其他实验仅使用随机数据集。对于实验。

表 1 基于 Transformer 的旅行商问题模型比较：Kool 等人 [4], TSPTransformer [5], Tspformer [25], 以及我们的模型。其中，MHSA 和 MMHA 分别表示多头自注意力和掩码多头注意力。Kool 等人的解码器未使用 MHSA。 [4]

模型	编码器			解码器			
	嵌入	MHSA		MHSA		MMHA	
		查询	Key	查询	Key	查询	Key
Kool 等人 [4]	线性	每个节点	每个节点	–	–	第一个节点 +最后一个节点 + 图	每个 未访问节点
TSP Transformer [5]	线性	每个节点	每个节点	最后一个节点	每个 已访问节点	最后一个节点	每个 未访问节点
Tspformer [25]	线性	采样节点	每个节点	–	–	最后一个节点	每个 未访问节点
Ours	CNN	每个节点	每个节点	最后一个节点	最后一个-m 已访问节点	最后一个节点	每个 未访问节点

1 和 2 中，我们采用贪心解码器来寻找最优路径。

实验1。 我们测试解码器中提出的部分自注意力，与标准 Transformer 解码器中现有的全连接自注意力相比，是否能更有效地提升模型性能。我们通过将参考向量的数量（m）从200减少到5来进行测试。

实验2. 我们移除了 CNN-Transformer 模型中的 CNN嵌入层，并通过消融研究检验该层在提取局部空间信息和提升输出性能方面是否有效。

实验3. 我们将所提出的模型与其他求解器进行比较，包括最优求解器（Concorde [6]）、启发式求解器（如 2-opt 搜索 [38], 蒙特卡洛树搜索（MCTS） [39],Google OR-Tools [3], 以及其他用于旅行商问题的最先进的基于 Transformer 的模型 [4, 5, 25, 30]。表1 总结了所提出的模型与其他基于 Transformer 的模型之间的主要差异。此处，“节点”指各模型的编码器或多头自注意力层生成的特定节点的隐藏特征向量，“图”指所有节点的隐藏特征向量的平均值。

实验 4. 我们比较所提出的 CNN-Transformer 模型与其他基于 Transformer 的模型的计算复杂度。

3.4 指标

使用以下指标评估模型的性能。

平均预测路径长度。设$\hat{}$ l_i^{TSP}为 i^{th}实例的预测路径长度。则平均路径长度计算为 $\frac{1}{n}\sum_{i=1}^{n}\hat{}l_i^{\mathrm{TSP}}$，其中 n 为测试实例总数。这里我们将 n 设为 10,000。

最优性差距。最优性差距的计算方式是预测路径长度相对于最优解的平均百分比，该最优解计算为

$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\tilde{l}_i^{\mathrm{TSP}}}{l_i^{\mathrm{TSP}}}-1\right)$，其中 n 是测试实例总数。此处， l_i^{TSP}是由 Concorde 生成的 i^{th} 实例的最优解 [6]。我们将 n 设为 10,000。

训练时间。 训练时间是指以512的批大小训练10,000个实例所需的时间。

推理时间。 我们报告了求解由100个TSP100和TSP200实例组成的测试集所需的推理时间。由于内存容量限制，束宽（B）设为2,500，批大小设为1。

GPU 内存使用量。 测量训练过程中的最大内存使用量。

4 结果

4.1 实验1

表2 展示了我们的模型在不同 m 取值下的平均路径长度和最优性差距。对于TSP150和TSP200，当 m 分别为100和20时，所提出的模型表现最佳。我们的实验结果表明，最优的 m 取值会随问题规模而变化。对于TSP100，采用 $m = 5$ 的部分自注意力的所提出模型实现了2.83%的最优性差距，而采用全连接注意力（即 $m = 100$）的所提出模型则产生了3.10%的最优性差距。

表 2 在随机数据集上，针对不同 TSP 实例规模，模型采用不同的 m （参考向量数量）时的平均路径长度（目标）和最优性差距（差距）。本实验使用贪心解码器。

	TSP50		TSP100		TSP150		TSP200	
m	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
200	–	–	–	–	–	–	12.417	15.97%
150	–	–	–	–	9.793	4.60%	12.634	18.00%
100	–	–	8.005	3.10%	**9.764**	**4.29**%	12.344	15.28%
50	5.750	1.06%	7.993	2.94%	9.778	4.44%	12.323	15.09%
20	5.748	1.01%	7.988	2.87%	9.779	4.45%	**12.004**	**12.11**%
5	**5.745**	**0.97**%	**7.985**	**2.83**%	9.773	4.38%	12.353	15.37%

4.2 实验2

表 3展示了消融研究的结果。两个模型均使用贪心解码器。我们的结果表明，对于所有TSP问题规模，所提出的模型均优于不含CNN嵌入层的模型。我们观察到，移除CNN嵌入层会降低整体性能。因此，我们可以得出结论

表3 消融研究。在随机数据集上，针对不同旅行商问题实例，比较不使用卷积神经网络和使用卷积神经网络的模型的平均路径长度（目标）和最优性差距（差距）。本实验使用贪心解码器

	TSP50		TSP100		TSP150		TSP200	
	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
Ours（不使用卷积神经网络）	5.754	1.12%	8.004	3.08%	9.811	4.79%	12.306	14.93%
Ours	**5.745**	**0.97**%	**7.985**	**2.83**%	**9.764**	**4.29**%	**12.004**	**12.11**%

我们的模型中的CNN嵌入层能够有效提取空间特征并提升输出性能。

4.3 实验3

随机数据集。表4 比较了所提出模型与其他最先进的基于Transformer的模型的性能。我们将结果分为三个部分展示：精确求解器、启发式方法和深度学习模型。Concorde [6],以能够得出最优结果而闻名，表现最佳。在深度学习模型中，所提出的模型在贪心算法和集束搜索解码方面均表现最佳。实验结果表明，我们的模型在性能和泛化能力方面显著优于蒙特卡洛树搜索 [39] 和 2-opt搜索方法 [38] 。对于 TSP50，与精确求解器 Concorde 相比，所提出模型的最优性差距仅高出 0.1%。与 TSP Transformer 相比，我们的模型采用束搜索后，TSP50 的最优性差距从 0.11% 降至 0.10%，TSP100 的最优性差距从 1.26% 降至 1.11%。

对于随机数据集，所提出的模型与 H-TSP 相比表现出具有竞争力的性能。与 H-TSP 相比，所提出的模型在 TSP50 和 TSP100 上表现更好，而 H-TSP 在 TSP 200 上表现更好。这是因为 H-TSP 被认为是专为大规模TSP设计的模型，在此类实例中表现出色。

TSPLIB 数据集。表5 展示了我们的模型和其他基于 Transformer 的模型在各种 TSPLIB实例上的输出性能。所有模型均在 TSP50 上训练。我们观察到，所提出的模型在各种 TSPLIB实例上均能取得稳定的结果，并且在大多数 TSPLIB实例上优于 TSP Transformer。因此，采用部分自注意力的所提出模型在真实世界数据集上同样有效。所提出的模型在所有 TSPLIB实例上均优于 H-TSP，并且除一种情况（即 rd100）外，在所有 TSPLIB实例上均优于 TSP Transformer。

TSPLIB中最难的两个实例是kroC100和berlin52。berlin52被认为是一个难解的TSP实例，因为许多节点在非常狭小的区域内受到高度约束。对于极难的真实世界实例，所提出的模型在参与比较的模型中表现最佳。图4展示了Concorde和各种模型在kroC100与berlin52上的输出。对于kroC100，Concorde的最优巡回长度为20,749。我们模型的巡回路线长度为21,523，而TSP Transformer 模型的巡回路线长度为21,788。对于berlin52，巡回路线长度

表4 使用随机数据集时，各种模型和求解器在不同旅行商问题实例规模下的平均路径长度（目标）和最优性差距（差距）。带 * 的结果引自其他论文。在 *type* 列中，H：启发式算法，RL：强化学习，G：贪心算法，S：采样，BS：束搜索

方法	Type	TSP50 Obj.	TSP50 Gap	TSP100 Obj.	TSP100 Gap	TSP150 Obj.	TSP150 Gap	TSP200 Obj.	TSP200 Gap
Concorde [6]	精确求解器	5.689	0.00%	7.764	0.00%	9.363	0.00%	10.708	0.00%
蒙特卡洛树搜索 [39]	H	7.323	28.72%	12.092	55.74%	17.230	84.02%	22.475	109.89%
2-opt 搜索 [38]	H	6.451	13.39%	9.180	18.24%	11.277	20.44%	13.084	22.19%
OR Tools [3]	H	5.863	3.06%	8.094	4.25%	9.826	4.94%	11.280	5.34%
Kool 等人 [4]	RL, G	5.795	1.86%	8.107	4.42%	9.938	6.14%	13.246	23.70%
	RL, S	5.727	0.67%	7.941	2.28%	9.709	3.70%	12.227	14.19%
TSP Transformer [5]	RL, G	5.750	1.05%	8.015	3.22%	9.814	4.83%	12.571	17.40%
	RL, BS	5.695	0.11%	7.863	1.26%	9.616	2.71%	12.135	13.33%
Tspformer [25]	RL, G	5.874	3.25%	8.175	5.29%	9.958	6.35%	13.371	24.87%
	RL, BS	5.726	0.65%	7.953	2.43%	9.706	3.66%	12.865	20.14%
H-TSP [30]	强化学习, G	5.748	1.04%	7.889	1.61%	**9.574**	**2.25%**	**10.990**	**2.63%**
Ours	RL, G	5.745	0.97%	7.985	2.83%	9.764	4.29%	12.004	12.11%
	RL, BS	**5.695**	**0.10%**	**7.851**	**1.11%**	9.589	2.42%	11.637	8.68%

表5 TSPLIB实例的巡回路线长度（目标）和最优性差距（差距）。关键参数 $\frac{l}{\sqrt{N \cdot A}}$ 接近 0.75，表明旅行商问题实例更难。本实验采用贪心解码器

问题	$\frac{l}{\sqrt{N \cdot A}}$	Concorde [6]	Kool 等人 [4] Obj.	Kool 等人 [4] Gap	TSP Transformer [5] Obj.	TSP Transformer [5] Gap	Tspformer [25] Obj.	Tspformer [25] Gap	H-TSP [30] Obj.	H-TSP [30] Gap	Ours Obj.	Ours Gap
kroC100	0.75	20,749	21,565	3.93%	21,788	5.01%	23,297	12.28%	22,046	6.25%	**21,523**	**3.73%**
berlin52	0.74	7,542	8,017	6.30%	7,637	1.26%	7,940	5.28%	7,718	2.33%	**7,610**	**0.90%**
kroA100	0.77	21,282	23,078	8.44%	21,747	2.18%	24,008	12.81%	22,758	6.94%	**21,620**	**1.59%**
ch150	0.78	6,528	7,242	10.94%	7,390	13.20%	7,274	11.43%	7,268	11.34%	**7,050**	**8.00%**
ch130	0.78	6,110	**6,549**	**7.18%**	6,569	7.51%	6,808	11.42%	6,601	8.04%	6,552	7.23%
rd100	0.81	7,910	8,441	6.71%	8,078	2.12%	8,549	8.08%	8,481	7.22%	**8,044**	**1.69%**
st70	0.86	675	698	3.41%	710	5.19%	701	3.85%	698	3.41%	**676**	**0.15%**
eil101	0.98	629	**665**	**5.72%**	681	8.27%	690	9.70%	677	7.63%	668	6.20%
eil76	1.03	538	**560**	**4.09%**	565	5.02%	592	10.04%	569	5.76%	564	4.83%
eil51	1.05	426	439	3.05%	438	2.82%	435	2.11%	438	2.82%	**429**	**0.70%**

我们模型的巡回路线长度为7,610，而TSP Transformer 模型的巡回路线长度为7,637。

4.4 实验 4

表6 总结了针对 TSP100 和 TSP200 的各种基于 Transformer 的模型的整体模型复杂度和运行时间。与其他模型相比，所提出的模型在 GPU 内存使用量、训练时间和推理时间方面具有竞争力。结果表明，在所比较的模型中，H-TSP 的计算复杂度最高。它需要通过预训练为低层模型额外设置一个预热阶段，并且训练过程相当复杂。因此，它的缺点是需要大量计算资源和较长的训练时间。

对于 TSP200，H-TSP 的模型参数数量是我们模型的 3.7 倍，训练时间约为我们模型的 68 倍。由于我们在模型中添加了 CNN 嵌入层，因此所提出的模型的模型参数数量多于 TSP Transformer。不过，我们的模型在

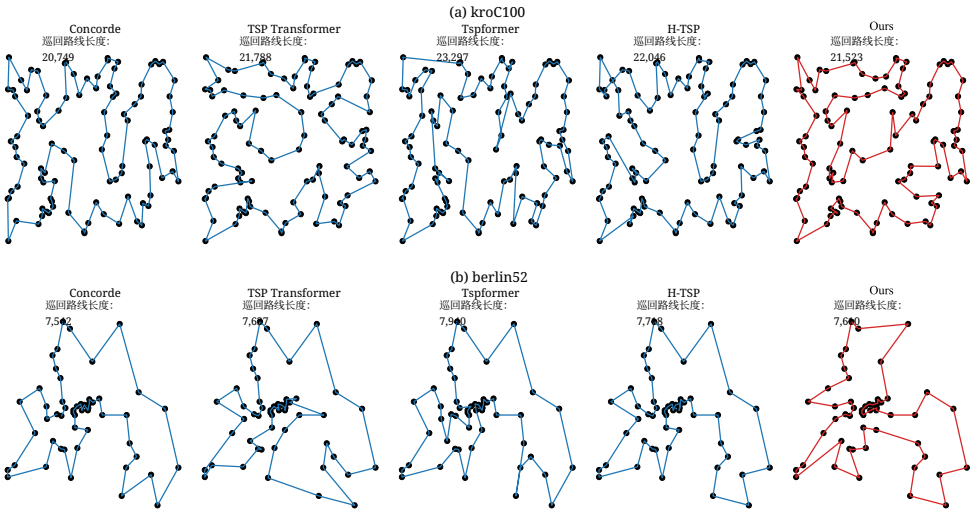


图 4 Concorde 的输出路径 [6], TSPTransformer [5], Tspformer [25], H-TSP [30], 以及我们的方法在使用 TSPLIB 数据集的 (a) kroC100 和 (b) berlin52 上的结果

表 6 TSP100 和 TSP200 的 10,000 个旅行商问题实例的模型参数、GPU 内存使用量和训练（T）时间，以及 100 个旅行商问题实例的推理（I）时间对比。G：贪心算法，BS：束搜索

问题 size	模型	参数量	GPU 内存使用量（GB）	预训练 time	T 时间	I 时间（G）	I 时间（基准方案）
TSP100	TSP Transformer [5]	141万	16.59	–	18.27 秒	10.05秒	1分15秒
	Tspformer [25]	1.08M	12.46	–	15.66秒	8.55秒	1分04秒
	H-TSP [30]	5.3M	20.12	1分12秒	21分37秒	8.99秒	–
	Ours	1.43M	13.63	–	18.08秒	11.47秒	51.70秒
TSP200	TSP Transformer [5]	1.41M	29.28	–	1m9s	19.67秒	4分50秒
	Tspformer [25]	1.08M	20.85	–	57.64秒	16.75秒	4分04秒
	H-TSP [30]	5.3M	22.04	5分40秒	76分43秒	17.95秒	–
	Ours	1.43M	23.48	–	1m8s	23.09秒	3分22秒

TSP100 和 TSP200 上分别少消耗 17.8% 和 19.8% 的 GPU 内存。与 Yang 等人提出的 Tspformer 模型相比， [25],我们的模型的计算复杂度略高。

5 讨论

近期研究尝试采用多种启发式搜索算法（如蒙特卡洛树搜索 [40, 41] 或 2-opt 搜索 [21, 22] ），以进一步提升TSP解的质量。所提出的模型采用束搜索解码技术，但也可以结合其他针对TSP的启发式搜索算法，进一步提升性能。我们还发现， [11] 提出的最短巡回启发式算法从B条完整巡回路线中选出最短的一条，也能提升模型的输出性能。

传统的基于优化的求解器（如 Concorde） [6], 在性能方面仍优于神经网络模型。然而，值得注意的是，神经

网络模型的推理速度比 Concorde 更快。例如，Concorde 求解 100 个 TSP100 旅行商问题实例约需 16 秒，而所提出的模型约需 11 秒。

我们的结果表明，所提出的模型能够显著降低内存消耗和推理时间。我们的模型也基于标准 Transformer 模型，包含多个 Transformer模块，并添加了一个 CNN嵌入层来提取空间特征。我们计划将文献 [26–29, 42] 中提出的各种轻量化技术应用于基于 Transformer 的模型，也计划将其应用于卷积神经网络 [43–46]。

基于星形拓扑结构的 Star-Transformer模型已被证明是适用于许多自然语言处理任务的一种有效轻量化技术 [26]。我们计划在 Star-Transformer模型使用的编码器中引入中继节点，以进一步提升性能。近期研究表明，ProbSparse自注意力也能有效减轻基于 Transformer 的模型的负担 [29]。我们还计划通过引入 ProbSparse自注意力进一步轻量化所提出的 CNN-Transformer 模型。最后，所提出的 CNN嵌入层性能优于线性嵌入层，但需要更高的计算复杂度。在后续研究中，我们计划利用 MobileNet [43, 44]中的轻量化技术，进一步轻量化 CNN嵌入层。

6 结论

我们提出了首个在解码器中采用部分自注意力的 CNN-Transformer 模型。该模型能够从输入数据中提取并学习空间特征，与标准基于 Transformer 的模型相比，可生成更好的 TSP解。结果表明，与标准 Transformer模型相比，我们的模型能够更好地学习局部组合性。我们还观察到，在解码器中采用部分自注意力，显著降低了模型的 GPU 内存使用量和推理时间。结果还表明，所提出的 CNN嵌入层和部分自注意力能够有效提升模型在随机数据集和真实世界数据集上的性能。我们发现，解码器中的全连接自注意力会降低性能；仅对最近访问过的节点执行部分自注意力，能够取得优于全连接注意力模型的表现。所提出的模型在多个方面均优于现有最先进的基于 Transformer的模型，并在真实世界数据集上展现出最佳性能。

致谢。本研究部分得到了韩国国家研究基金会（NRF）资助，该基金由韩国政府（科学技术信息通信部）提供，项目编号为 NRF-2022R1A4A5034121。本研究还部分得到了韩国科学技术信息通信部通过 ICAN（信息通信技术挑战与高级人才培养网络）支持计划（IITP-2024-RS-2023-00260175）提供的资助，该计划由信息通信企划评价院监督。

声明

利益冲突。作者声明，他们在本研究中不存在任何利益冲突。

所用数据的伦理与知情同意。不适用。

作者贡献。**Minseop Jung：**软件、方法学、研究，**Jaeseung Lee：**软件、验证，**Jibum Kim：**撰写——初稿、撰写——审阅与编辑、监督。

数据可用性。数据集和代码可在以下网址获取https://github.com/cm8908/CNN Transformer3。-

参考文献

[1]Papadimitriou CH (1977) 欧几里得旅行商问题是NP完全的。《理论计算机科学》4(3):237–244[2] Christofides N (1976) 旅行商问题中新启发式算法的最坏情况分析。技术报告[3]Perron L, Furnon V (2022) OR-Tools。网址 https://developers.google.com/optimization/，2022-11-25[4] Kool W, Van Hoof H, Welling M (2018) 注意力，学会解决路径规划问题！arXiv:1803.08475[5] Bresson X, Laurent T (2021) 用于旅行商问题的Transformer网络。arXiv:2103.03012[6] Applegate D, Bixby R, Chv´atal V, 等 (2006) Concorde TSP求解器。网址 https://www.math.uwaterloo.ca/tsp/concorde/[7] Vinyals O, Fortunato M, Jaitly N (2015) 指针网络。《神经信息处理系统进展》28[8] Gurobi Optimization, LLC (2023) 《Gurobi优化器参考手册》。网址 https://www.gurobi.com[9] Bello I, Pham H, Le QV, 等 (2016) 使用强化学习进行神经组合优化。arXiv:1611.09940[10] Nazari M, Oroojlooy A, Snyder L, 等 (2018) 用强化学习解决车辆路径问题。《神经信息处理系统进展》31[11] Joshi CK, Laurent T, Bresson X (2019) 用于旅行商问题的高效图卷积网络技术。arXiv:1906.01227

[12]Stohy A, Abdelhakam HT, Ali S, 等 (2021) 用于旅行商问题优化的混合指针网络。《PLOS ONE》16(12):e0260995[13] Ma Q, Ge S, He D, 等 (2019) 通过图指针网络和分层强化学习进行组合优化。arXiv:1911.04936[14] Miki S, Ebara H (2019) 使用基于图像的分类解决旅行商问题。载于：2019 IEEE 第31届人工智能工具国际会议（ICTAI）。IEEE，第1118–1123页[15] Ling Z, Tao X, Zhang Y, 等 (2020) 通过全卷积网络解决优化问题：旅行商问题的应用。IEEE《系统、人和控制论汇刊：系统》51(12):7475–7485[16] Sultana N, Chan J, Sarwar T, 等 (2022) 学习优化一般旅行商问题实例。《国际机器学习与控制论杂志》，第1–16页[17] Vaswani A, Shazeer N, Parmar N, 等 (2017) Attention Is All You Need。《神经信息处理系统进展》30[18] Dai Z, Yang Z, Yang Y, 等 (2019) Transformer-XL：超越固定长度上下文的注意力语言模型。arXiv:1901.02860[19] Devlin J, Chang MW, Lee K, 等 (2018) BERT：用于语言理解的深度双向变换器预训练。arXiv:1810.04805[20] Dosovitskiy A, Beyer L, Kolesnikov A, 等 (2020) 一幅图像胜过16×16个词：用于大规模图像识别的Transformer。arXiv:2010.11929[21] Deudon M, Cournut P, Lacoste A, 等 (2018) 通过策略梯度学习TSP启发式算法。载于：约束规划、人工智能与运筹学集成国际会议。施普林格，第170–181页[22] Wu Y, Song W, Cao Z, 等 (2021) 学习用于解决路径规划问题的改进启发式方法。IEEE《神经网络与学习系统汇刊》[23] Kwon YD, Choo J, Kim B, 等 (2020) POMO：具有多个最优解的强化学习策略优化。《神经信息处理系统进展》33:21188–21198[24] Goh YL, Lee WS, Bresson X, 等 (2022) 结合强化学习与最优传输解决旅行商问题。arXiv:2203.00903[25]Yang H, Zhao M, Yuan L, 等 (2023) 用于旅行商问题的内存高效基于Transformer的网络模型。《神经网络》161:589–597[26] Guo Q, Qiu X, Liu P, 等 (2019) Star-Transformer。arXiv:1902.09113

[27] Beltagy I、Peters ME、Cohan A（2020）Longformer：用于长文档的 Transformer。arXiv:2004.05150[28] Wang S、Li BZ、Khabsa M 等（2020）Linformer：具有线性复杂度的自注意力。arXiv:2006.04768[29] Zhou H、Zhang S、Peng J 等（2021）Informer：超越高效 Transformer 的长序列时间序列预测。在：人工智能促进协会年会论文集，第 11106–11115 页[30] Pan X、Jin Y、Ding Y 等（2023）H-TSP：分层求解大规模旅行商问题。在：AAAI 2023，网址
h
t
t
p
s://
w
w
w.
m
i
c
r
o
s
o
f
t.
c
o
m/
e
n-us/research/publication/h-tsp-hierarchically-solving-the-large-scale-traveling-sales
man-problem/[31] Ren H、Zhong R、Gui H（2023）一种基于自比较的动态旅行商问题强化学习方法。SSRN 在线资源：https://dx.doi.org/10.2139/ssrn.4643703[32] He K、Zhang X、Ren S 等（2016）用于图像识别的深度残差学习。在：IEEE计算机视觉与模式识别会议论文集，第 770–778 页[33] Ioffe S、Szegedy C（2015）批归一化：通过减少内部协变量偏移加速深度网络训练。在：国际机器学习大会。PMLR，第 448–456 页[34] Reinelt G（1991）TSPLIB——旅行商问题库。《ORSA计算期刊》3(4)：376–384[35] Gent IP、Walsh T（1996）TSP相变。《人工智能》88(1-2)：349–358[36] C´ardenas-Montes M（2018）创建难解的旅行商问题实例。《应用软计算》71：268–276[37] Lowerre BT（1976）Harpy语音识别系统[博士学位论文]。卡内基梅隆大学[38] Croes GA（1958）一种求解旅行商问题的方法。《运筹学》6(6)：791–812。网址
http://www.jstor.org/stable/167074

[39] Kocsis L, Szepesv´ari C (2006) 基于老虎机算法的蒙特卡洛规划。《机器学习：ECML 2006》。施普林格柏林海德堡出版社，柏林，海德堡，第282–293页[40] Xing Z, Tu S (2020) 一种由图神经网络辅助的蒙特卡洛树搜索旅行商问题求解方法。IEEE Access 8:108418–108428[41] Xing Z, Tu S, Xu L (2020) 利用蒙特卡洛树搜索和深度神经网络求解旅行商问题。arXiv:2005.06879[42] Mehta S, Ghazvininejad M, Iyer S, 等 (2020) Delight：深度轻量级 Transformer。arXiv:2008.00623[43] Howard AG, Zhu M, Chen B, 等 (2017) MobileNets：适用于移动视觉应用的高效卷积神经网络。arXiv:1704.04861[44] Sandler M, Howard A, Zhu M, 等 (2018) MobileNetV2：倒置残差与线性瓶颈结构。载于：IEEE计算机视觉与模式识别会议论文集，第4510–4520页[45] Zhang X, Zhou X, Lin M, 等 (2018) ShuffleNet：一种适用于移动设备的超高效卷积神经网络。载于：IEEE计算机视觉与模式识别会议论文集，第6848–6856页[46] Ma N, Zhang X, Zheng HT, 等 (2018) ShuffleNet V2：高效 CNN 架构设计的实用指南。载于：欧洲计算机视觉会议（ECCV）论文集，第116–131页