表示具有全局注意力的图神经网络的远程上下文

Zhanghao Wu*[1]、Paras Jain*[1]、Matthew A. Wright[1]Azalia Mirhoseini[2]、Joseph E. Gonzalez[1]、Ion Stoica[1] [1]UC Berkeley、
谷歌大脑
通讯至：{zhwu, paras_jain}@berkeley.edu
*同等贡献，通过随机掷硬币确定。

摘要

图神经网络是结构化数据集的强大架构。然而，当前的方法难以表示长期依赖关系。缩放 GNN 的深度或宽度不足以扩大感受野，因为较大的 GNN 会遇到优化不稳定性，例如梯度消失和表示过度平滑，而基于池的方法尚未像计算机视觉那样普遍有用。在这项工作中，我们建议使用基于 Transformer 的自注意力来学习远程成对关系，并使用一种新颖的“读出”机制来获得全局图嵌入。受最近的计算机视觉结果的启发，发现位置不变注意力在学习远程关系方面表现出色，我们称为 GraphTrans 的方法在标准 GNN 模块之后应用了置换不变的 Transformer 模块。这种简单的架构在几个图分类任务上产生了最先进的结果，优于显式编码图结构的方法。我们的结果表明，没有图结构的纯基于学习的方法可能适用于学习图上的高级、长期关系。GraphTrans 的代码可在 https://github.com/ucbrise/graphtrans.

1 介绍

图神经网络（GNN）使深度网络能够处理结构化输入，例如分子或社交网络。GNN 学习映射，从其邻域的结构和特征计算图节点和\/或边缘的表示。这种邻域局部聚合利用了由图的连通性编码的关系归纳偏差[3]。与卷积神经网络（CNN）类似，GNN 可以通过堆叠层来聚合来自本地邻域之外的信息，从而有效地拓宽了 GNN 的接受域。

然而，据观察，当 GNN 的深度超过几层时，GNN 的性能会急剧下降。这种限制损害了 GNN 在全图分类和回归任务中的性能，我们希望预测描述整个图的目标值，该目标值可能依赖于接收域有限的 GNN 可能无法捕获的远程依赖关系[35]。例如，考虑一个大图，其中节点 A 必须关注距离 K 跳远的节点 B。如果我们的 GNN 层仅在节点的单跳邻域上聚合，则需要 K 层 GNN。然而，这个 GNN 的感受野的宽度将呈指数增长，稀释了来自节点 B 的信号。也就是说，简单地将感受野扩展到 K-hop 邻域也可能无法捕捉到这些长期依赖关系[40]。通常，“太深”的 GNN 会导致节点表示在整个图上崩溃以等效，这种现象有时称为过度平滑或过度挤压[21, 5, 2]。总而言之，常见 GNN 架构的最大上下文大小受到了有效限制。

第 35 届神经信息处理系统会议（NeurIPS 2021），澳大利亚悉尼。

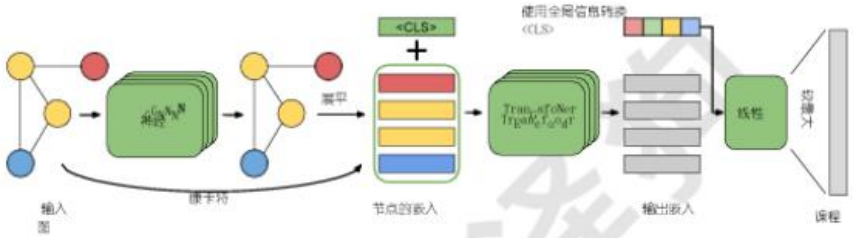


图 1：GraphTrans 的架构。一个标准的 GNN 子模块学习局部的、短程的结构，然后一个全局的 Transformer 子模块学习全局的、长程的关系。

许多作者提出通过类似于当今 CNN 中发现的中间池操作来解决过度平滑问题。图池化操作逐渐粗化渐进式 GNN 层中的图，通常通过将邻域折叠成单个节点 [9, 37, 20, 等等。]。理论上，层次粗化应该允许更好的远程学习，既可以减少信息必须传播的距离，也可以过滤掉不重要的节点。然而，到目前为止，还没有发现像 CNN 池化一样普遍适用的图池化操作。最先进的结果通常是通过不使用中间图粗化的模型获得的[27]，一些结果表明邻域局部粗化可能是不必要的或适得其反[23].

在这项工作中，我们在图池化和学习 GNN 中的长期依赖关系方面采用了不同的方法。与分层池化一样，我们的方法也受到计算机视觉方法的启发：我们将一些明确编码相关关系归纳偏差的原子操作（即 CNN 中的卷积或空间池化，GNN 中的邻域粗化）替换为注意力等纯学习操作[11, 4, 7].

我们的方法，我们称之为 Graph Transformer（GraphTrans，见图 1）。1），在标准 GNN 层堆栈之上添加了一个 Transformer 子网络。这个 Transformer 子网以与位置无关的方式显式计算所有成对节点交互。这种方法很直观，因为它保留了 GNN 作为一种专门的架构来学习节点直接邻域结构的局部表示，同时利用 Transformer 作为强大的全局推理模块。这与最近的计算机视觉架构相似，其中作者发现硬关系归纳偏差对于学习短程模式很重要，但在建模长程依赖关系时用处不大甚至适得其反[25]。由于没有位置编码的 Transformer 是排列不变的，我们发现它很适合图。此外，GraphTrans 不需要任何专门的模块或架构，并且可以在任何现有 GNN 主干之上的任何框架中实现。

我们在各种流行的图分类数据集上评估 GraphTrans。我们发现 OpenGraphBenchmark [15] 我们在两个图分类任务上取得了最先进的结果。此外，我们发现分子数据集 NCI1 有了实质性的改进。令人惊讶的是，我们发现我们的简单模型通过层次聚类（例如自注意力池）在图中的远程建模中优于复杂基线 [20].

我们的贡献如下：

- 我们展示了通过 Transformers 进行的远程推理提高了图神经网络（GNN）的准确性。我们的结果表明，对图中所有成对的节点-节点交互进行建模对于大型图分类任务尤为重要。
- 我们引入了一种新颖的 GNN“读出模块”。受 Transformers 的文本分类应用的启发，我们使用了一个特殊的“<CLS>”令牌，其输出嵌入将所有成对的交互聚合到一个单一的分类向量中。我们发现这种方法优于全局池化等非学习读取方法以及图形特定池化方法等学习聚合方法[37, 20] 和“虚拟节点”方法。
- 使用我们新颖的架构 GraphTrans，我们在几个 OpenGraphBenchmark [15] 数据集和 NCI 生物分子数据集 [30].

2 相关工作

图分类。图分类是实际应用中的一项重要任务。尽管 GNN 将结构化数据编码为节点表示，但将表示聚合到单个图嵌入以进行图分类仍然是一个问题。与 CNN 类似，GNN 中的池化可以是全局的，将一组节点和\/或边编码减少为单个图形编码，也可以是局部的，折叠节点和\/或边的子集以创建更粗略的图形。在 CNN 中并行使用中间池，一些作者提出了局部池操作，旨在用于 GNN 层堆栈，逐步粗化图。提出的方法包括学习池化方案[37, 20, 14, 16, 1 等]和基于经典图粗化方案的非学习池化方法[10, 9, 等等。]。然而，在 GNN 中分层、基于粗化的池化的有效性或必要性尚不清楚 [23]。另一方面，最常见的全局全图池化方法是 i）节点上的非学习均值或最大池化和 ii）“虚拟节点”方法，其中最终 GNN 层输出单个嵌入连接到图中每个“真实”节点的虚拟节点。

与图池化相关的一项值得注意的工作是 Thost 和 Chen 的 DAGNN（有向无环图神经网络）[27]，它在 OGBG-Code2 上获得了之前最先进的精度。 DAGNN 层通过遍历 DAG 的 RNN 聚合每一层内的整个图，这与大多数 GNN 层仅聚合在节点的邻域上不同。虽然他们没有将此方法描述为池化操作，但它与 GraphTrans 相似，因为它充当学习的全局池化（因为它将 DAG 中每个节点的嵌入聚合到汇节点中），可以对远程建模依赖关系。请注意，GraphTrans 也是 DAGNN 的补充，因为它们的最终图级池化操作是在汇节点上的全局最大池化，而不是学习操作。

图上的变形金刚。几位作者研究了 Transformer 架构在图上的应用。最近的作品，如张等人。 [38]，荣等人。 [24]，以及 Dwivedi 和 Bresson [12] 提出 GNN 层，让节点通过 Transformer 风格的注意力关注一些周围邻域中的其他节点，而我们使用自注意力来进行置换不变的图级池化或“读出”操作，将节点编码折叠为单个图编码.其中，张等人。 [38] 和荣等人。 [24] 通过允许节点参与的不仅仅是单跳邻域，解决了在不过度平滑的情况下学习远程依赖的问题：Zhang 等人。 [38] 将参与的邻域半径作为调整参数，Rong 等人。 [24] 在训练和推理期间关注随机大小的邻域。相比之下，我们使用全图自注意力来允许学习长期依赖关系。

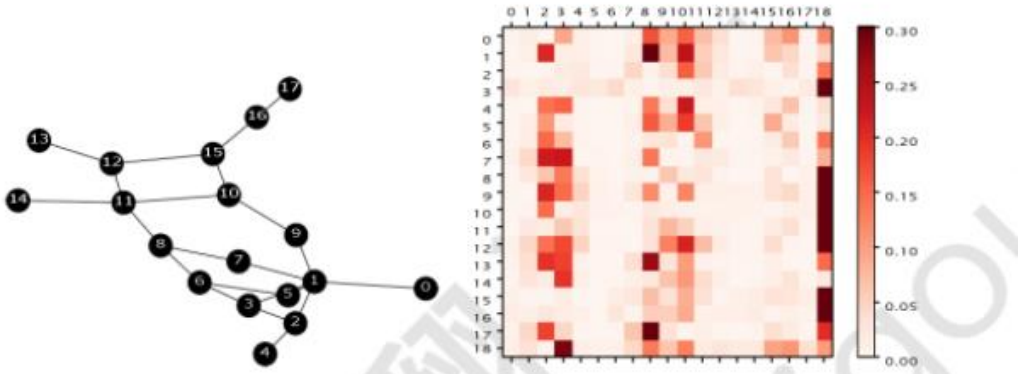
而张等人。 [38] 不考虑全图预测问题，例如 Dwivedi 和 Bresson [12]，当图分类或回归需要全图嵌入时，他们在节点上使用全局平均池化，而 Rong 等人。 [24] 对节点进行加权求和，计算的权重绕过 h^l到两层 MLP。另请注意，先前的工作考虑了 Transformer 位置编码的特定于图形的版本，而我们省略了位置编码以确保排列不变性。

高效的变压器。变压器 [28]已广泛用于序列建模。最近，出现了对变压器架构的修改，以进一步提高效率[34, 19, 6]。 LiteTransformer [34] FLOPs 更少，Reformer [19] 具有复杂性，而执行者 [6] 具有较少的计算和内存复杂性。神经架构搜索（NAS）也被应用于 Transformer 来满足边缘设备的资源约束 [32]。这些现成的架构与我们的 GraphTrans 正交，可用于提高可扩展性。

3 动机：学习对图上的远程成对交互进行建模

总而言之，尝试通过堆叠 GNN 层或分层池化在图上进行远程学习尚未导致性能提升，虽然一些工作已经显示出将单个 GNN 层的感受野扩展到单跳邻域之外的一些成功[38, 24, 40]，这种方法将如何扩展到具有数千个节点的非常大的图还有待观察。

在最近的计算机视觉文献中可以找到替代方法的灵感。在过去的几年里，研究人员发现注意力机制可以作为替代品



(a) 来自 Code2 的示例图。 (b) 来自 GraphTrans 的相应注意力图。

图 2：GraphTrans 中的示例图和注意力图。该图是从 Code2 验证集中随机抽取的。注意力图是从我们的 GraphTrans 中的转换器模块的第一层检索的。水平轴对应于目标，水平轴对应于源（因此，注意力权重在水平轴上的总和为 1）。请注意，在 (b) 中，索引 18 对应于第 1 节中描述的特殊 〈CLS〉 标记4.

对于传统的 CNN 卷积 [4, 7]：注意力层可以学习重现由局部卷积引起的强关系归纳偏差。最近，一些计算机视觉任务的最先进的方法在传统的 CNN 主干之上使用了注意力样式的子模块 [2, 33, 等等。]。这些结果表明，虽然强关系归纳偏差有助于学习局部的、短期的相关性，但对于长期相关性来说，结构较少的模块可能是首选[2].

我们通过我们的 GraphTrans 模型将这种洞察力运用到图学习领域，该模型使用传统的 GNN 子网络作为主干，但将学习远程依赖关系留给没有图空间先验的 Transformer 子网络。如前所述，我们的 Transformer 应用程序让每个节点都关注每个其他节点（与将 Transformer 应用于仅允许关注邻域的图的其他方法不同），这激励 Transformer 学习最重要的节点-节点关系，而不是偏爱附近的节点（后一个任务已卸载到前面的 GNN 模块）。

定性地说，这个方案提供了长期关系确实很重要的证据。 GraphTrans 在 OGB Code2 数据集上的示例应用程序如图所示。2. 在这个任务中，我们采用通过解析 Python 方法获得的抽象句子树，并需要预测形成方法名称的标记。注意力图展示了与 Transformers 的 NLP 应用中发现的相似的模式：一些节点从许多其他节点接收到显着的权重，而不管它们之间的距离如何。请注意，节点 17 对节点 8 赋予了重要意义，尽管这两个节点相距 5 跳。另外，如图2的注意力图，索引 18 指的是对应于我们用作读出机制的特殊 〈CLS〉 标记的嵌入，下面将更详细地描述。我们允许这种嵌入是可学习的，因此参与它的许多节点（由第 18 列中的许多暗单元表示）可能表明这些节点正在从学习的嵌入中获得一些图通用记忆。这种定性可视化以及我们最新的最新结果表明，在学习远程依赖关系时移除空间先验对于有效的图摘要可能是必要的。

接下来讨论 GraphTrans 的实现细节。

4 使用 GraphTrans 学习全局信息

回看图1，GraphTrans 由两个主要模块组成：GNN 子网和 Transformer 子网。我们接下来详细讨论这些。

GNN 模块。我们考虑图属性预测，即对于每个图 = (,)，我们有一个图特定的预测目标 y 。我们假设每个节点 v 都有一个初始特征向量 $h^0 \in R^{d0}$。由于 GraphTrans 是一个普遍适用的框架，可以与各种 GNN 一起使用，我们对输入到 Transformer 子网络的 GNN 层做了很少的假设。一个通用的 GNN 层堆栈可以表示为

$$\boldsymbol{h}_v = f\left(\boldsymbol{h}_v^{l-1}, \{\boldsymbol{h}_u^{l-1} | u \in N(v)\}\right), \quad l = 1, \ldots, L_{\text{神经网络}} \tag{1}$$

其中 $L_{\text{神经网络}}$是 GNN 层的总数，(v) 是 v 的某个邻域，f(●) 是由神经网络参数化的某个函数。请注意，许多 GNN 层都承认边缘特征，但为了避免符号混乱，我们在这里省略了对它们的讨论。

变压器模块。一旦我们有了最终的每个节点的 GNN 编码 $h^{\text{新维度}}$，我们将它们传递给 GraphTrans 的 Transformer 子网络。Transformer 子网的运行方式如下。我们首先执行 $h^{\text{新维度}}$到 Transformer 维度的线性投影和层归一化
规范化嵌入：

$$h^0 = \text{LayerNorm}\left(W^{\text{输出}} h^{\text{新维度}}\right) \tag{2}$$

其中 $W^{\text{输出}} \in R^{dTF \times dL}$GNN 是可学习的权重矩阵，$d_{\text{特征团队}}$和 $d_{\text{新维度}}$分别是 Transformer 维度和最终 GNN 嵌入的维度。然后将投影的节点嵌入 h^0输入标准的 Transformer 层堆栈，没有附加的位置嵌入，因为我们期望 GNN 已经将结构信息编码到节点嵌入中：

$$a_{v,u} = \left(W^q h_v^{l-1}\right)^T \left(W^k h_u^{l-1}\right) / \sqrt{d_{\text{特征团队}}} \qquad a_{v,u} = \text{软最大}_{v,w}(a_{v,u})$$
$$\bar{\boldsymbol{h}}_v = \sum_{w \in V} a_{v,w} \boldsymbol{W}^V \boldsymbol{h}_w^{l-1} \tag{3}$$

其中 W^q, W^k, $W^v \in R^{dTF/nhead \times dTF/nhead}$是学习的查询、键和值矩阵，分别为
tively，对于 layer 中的单个注意力头。按照标准，我们运行 n_{head} 个并行注意力头并连接得到的每个头编码 h_v。然后将级联编码传递到 Transformer 全连接子网络，由标准 Dropout 组成 层范数 FC非线性 退出 → 足球俱乐部→ 退出 →ayer Nor⇒ 序列，残差连接从 h^{l-1}到第一个 dropout 之后，从第一个全连接子层之前到第一个 dropout 之后在第二个全连接子层之后立即退出。

<CLS> 嵌入作为 GNN“读出”方法。如前所述，对于全图分类，我们需要一个描述整个图的嵌入向量。在 GNN 文献中，这个将每个节点和/或边缘的嵌入折叠为单个嵌入的模块称为“读出”模块，最常见的读出模块是简单的均值或最大池化，或单个“虚拟节点”连接到网络中的每个其他节点。

在这项工作中，我们提出了一个类似于 Transformers 其他应用程序中使用的特殊令牌读出模块。在使用 Transformers 的文本分类任务中，一种常见的做法是在输入序列中附加一个特殊的 <CLS> 标记，然后再将其传递到网络中，然后将与该标记位置对应的输出嵌入作为整个句子的表示。通过这种方式，Transformer 将被训练以将句子的信息聚合到该嵌入，通过使用注意模块计算 <CLS> 标记与句子中的每个其他标记之间的一对一关系。

我们对特殊令牌读出的应用与此类似。具体来说，当输入转换后的每个节点嵌入 $\bar{O}$时，我们在序列中附加了一个额外的可学习嵌入 h_{CLS}，并以转换器输出的第一个嵌入 <CLS> $R^{\text{新维度}}$作为表示整个图（请注意，由于我们不包括位置编码，因此将特殊标记放置在句子的“开头”没有特殊的计算意义；该位置是按惯例选择的）。最后，我们应用线性投影，然后使用 softmax 来生成预测：

$$y = \text{最大软}\left(W^{\text{输出}} \bar{h}^{\text{新维度}}_{\text{<CLS>}}\right). \tag{4}$$

其中 $L_{\text{特征团队}}$是 Transformer 层的数量。

这种特殊令牌读出机制可以被视为虚拟节点读出的泛化或"深度"版本。虽然虚拟节点方法要求图中的每个节点都将其信息发送到虚拟节点，并且不允许学习图节点之间的成对关系，除非在虚拟节点的嵌入内（可能会产生信息瓶颈），但 Transformer 风格的特殊 -令牌读出方法让网络在需要在后面的层中提取它们之前，在早期层中学习远程节点到节点的关系。

5 实验

我们从三种模式评估 GraphTrans 的图分类任务：生物学、计算机编程和化学。我们的 GraphTrans 在所有这些基准上都取得了一致的改进，表明了框架的通用性和有效性。我们所有的模型都使用 Adam 优化器进行训练 [17]，学习率为 0.0001，权重衰减为 0.0001，默认 Adam β 参数。我们实验中使用的所有 Transformer 模块在前馈子网络中的嵌入维度 $d_{\text{前馈层}}$为 128，隐藏维度为 512。下面描述的 Transformer 基线仅使用节点嵌入序列进行训练，丢弃了图结构。

5.1 生物基准

数据集。我们选择了两个常用的图分类基准，NCI1 和 NCI109 [31]。它们每个都包含大约 4000 个图表，平均大约有 30 个节点，代表生化化合物。任务是预测一种化合物是否含有抗肺癌活性。我们按照[中的设置20, 2] 对于 NCI1 和 NCI109，以 8:1:1 的比例将数据集随机分成训练集、验证集和测试集。

训练设置。我们在 NCI1 和 NCI109 数据集上训练 GraphTrans 100 个 epoch，批量大小为 256。我们使用不同的随机种子运行每个实验 20 次，并计算测试精度的平均值和标准偏差。所有模型都遵循图中的架构1，具有 4 个变压器层，GNN 和 Transformer 模块的 dropout 比均为 0.1。我们使用先前文献中采用的两种不同设置来设置 GraphTrans 中 GNN 子模块的宽度和深度。小\ GraphTrans 模型中的 GNN 模块宽度和深度是从简单基线复制的，即 [20]，它的隐藏维度为 128 和 3 个 GNN 层。大 GraphTrans 模型中 GNN 模块的设置取自 OGB 提供的默认 GCN/GIN 模型，其隐藏维度为 300，GNN 层数为 4。我们还采用余弦退火时间表 [22] 用于学习率衰减。

结果。我们在表中报告了 NCI1 和 NCI109 的结果1. 简单的基线，包括 GCN Set2Set、SortPool 和 SAGPool，取自 [20]，而强基线 [13]，以及 FA 层 [2]。在表中1，我们的 Graph Transformer（小\）具有与简单基线相同的架构，但 NCI1 的平均准确度提高了 7.1%，NCI109 的平均准确度提高了 5.1%。我们还测试了以 GIN 作为编码器（GraphTrans（大））的框架，以与强基线中的设置保持一致，这也显著提高了 NCI1 的强基线的准确性 1.1% 和 NCI109 的 8.2%，即使没有深度 GNN，使用 4 层而不是 8 层。

5.2 化学基准

数据集。对于化学基准，我们在比 NCI 数据集更大的数据集上评估我们的 GraphTrans，来自 Open Graph Benchmark (OGB) 的 molpcba [15]。它包含 437929 个图，平均有 28 个节点。数据集中的每个图代表一个分子，其中节点和边分别是原子和化学键。任务是预测分子的多种特性。我们使用基准的标准拆分。 GIN 和 GIN-Virtual 基线的性能与 OGB 排行榜上的报告一致 [15].

训练设置。实验中的所有 GNN 模块都遵循 OGB 中提供的默认 GIN 模型的设置，具有 4 层和 300 个隐藏维度。我们以 256 的批大小\训练所有模型 100 个 epoch，并以最佳验证 ROC-AUC 报告测试结果。对彼此而言

表 1：NCI 生物数据集上的测试集精度。 GraphTrans 优于具有 FA 层的最先进的强基线。

模型	GNN 类型	GNN 层数	NCI1 (%)	NCI109 (%)
集合2集合[29, 20]	GCN	3	68.8±1.9	69.8±1.2
排序池 [39, 20]	GCN	3		74.0±1.2
SAGPool$_h$[20]	GCN	3	67.5±1.1	67.9±1.4
SAGPool$_g$[20]	全球通讯网络	3	81.5±1.2	74.1±0.8
埃里卡等人。(小)	杜松子酒	8	80.9±1.9	79.2±2.2
GraphTrans (小)	全球网络	3	81.9±1.4	79.2±2.2
GraphTrans (大)	杜松子酒	4	82.6±1.2	82.3±2.6

表 2：OGBG-Molpcba 数据集的结果。括号中的模型表示 GraphTrans 中 GNN 模块的 GNN 类型。总体而言，GraphTrans 优于竞争基线。

模型	有效 ROC-AUC	测试 ROC-AUC
全球通讯网络[18]	0.2059	0.2020
杜松子酒 [36]	0.2305	0.2266
GCN-虚拟 [18]	0.2495	0.2424
GIN-虚拟 [36]	0.2798	0.2703
变压器 [28]	0.1347	0.1304
GraphTrans (杜松子酒)	**0.2876**	**0.2721**
GraphTrans (GIN-虚拟)	**0.2858**	**0.2815**

GNN 和 Transformer 模块，我们应用了 0.3 的 dropout。我们使用 GIN 作为基线和 GNN 模块，因为它在 Molpcba 数据集上的性能优于 GCN 模型。

结果。在表中2，我们报告了关于 Molpcba 的验证和测试集的 ROC-AUC。虽然 Transformer 单独在这个数据集上效果很差，但我们的 GraphTrans 仍然提高了 GIN 和 GIN-Virtual 基线的 ROC-AUC。这表明我们的设计可以同时受益于 GNN 学习的局部图结构和 Transformer 模块基于 GNN 嵌入检索到的远程概念。

5.3 计算机编程基准

数据集。对于计算机编程基准，我们还采用了来自 OGB 的大型数据集 code2，它有 45741 个图，每个图平均有 125 个节点。该数据集是来自大约 450k Python 方法定义的抽象语法树 (AST) 的集合。任务是在给定由 AST 表示的方法体的情况下预测形成方法名称的子标记。我们还采用了从基准测试中分离出来的标准数据集。所有基线表现均在 OGB 排行榜上报告。

训练设置。我们还为来自 OGB 的 Code2 应用了 GCN 的默认设置，具有 4 个 GNN 层、300 个隐藏维度和 0.0 的 dropout 比率。我们对 Transformer 模块应用 0.3 的 dropout ratio 以避免过度拟合。由于数据集的规模很大，我们将所有模型训练了 30 个 epoch，批量大小为 16。对于 GraphTrans (PNA) 模型，我们遵循 [26]，GNN 模块的隐藏嵌入为 272，权重衰减为 3e-6。唯一的区别是我们仍然使用 0.0001 的学习率，而不是经过大量调整的 0.00063096 [26]。

表 3：OGBG-Code2 数据集的结果。所有基线都是从 OGB 排行榜收集的。 GraphTrans 优于最先进的 DAGNN。基于 PNA 模型的改进表明我们的方法与 GNN 模块的类型正交。

模型	有效的 F1 分数	测试 F1 分数
杜松子酒 [36]	0.1376	0.1495
全球通讯网络[18]	0.1399	0.1507
GIN-虚拟 [36]	0.1581	0.1439
GCN-虚拟 [18]	0.1461	0.1629
PNA公司[8]	0.1442	0.1585
达格恩 (SOTA) [27]	0.1607	0.1751
变压器 [28]	0.1539	0.1660
GraphTrans (GCN)	0.1594	0.1748
GraphTrans (GCN-虚拟)	**0.1670**	**0.1810**
GraphTrans (PNA)	**0.1698**	**0.1819**

表 4：OGBG-Code2 数据集的消融研究。仅使用冻结的预训练 GNN 模块训练 GraphTrans 中的 Transformer 模块也可以提高 F1 分数。这表明在 GNN 嵌入上训练 Transformer 可以学习 GNN 无法捕获的信息。

模型	有效的 F1 分数	测试 F1 分数
预训练的 GCN-Virtual	0.1457	0.1574
GraphTrans，预训练的 GCN-Virtual，冻结的 GNN	0.1479	0.1616
GraphTrans，预训练的 GCN-Virtual，微调的 GNN	**0.1564**	**0.1733**

结果。在表中3，我们将我们的 GraphTrans 与 Code2 数据集排行榜上的顶级架构进行比较。随着每个图中平均节点数的增加，全局信息变得更加重要，因为 GNN 越来越难以从远处的节点收集信息。即使没有大量调整，GraphTrans 也明显优于最先进的 (DAGNN) [27] 在排行榜上。

我们还包括 PNA 模型和我们的 GraphTrans 的结果，其中 PNA 模型作为 GNN 编码器。我们的 GraphTrans 也显着改进了结果，这表明我们的架构与 GNN 编码器模块的变体正交。

5.4 变形金刚可以捕捉长期关系

正如我们之前在图中观察到的2 并在部分讨论3，transformer 模块内部的注意力可以捕获 GNN 模块难以学习的远程信息。

为了进一步验证假设，我们设计了一个实验来证明 Transformer 模块可以学习到 GNN 模块的额外信息。在表中4，我们首先预训练一个 GNN（GCN-Virtual）直到收敛到 Code2 数据集，然后冻结 GNN 模型并在其后插入我们的 Transformer 模块。通过使用固定的 GNN 模块在训练集上训练模型，我们仍然可以观察到验证集的 F1 分数提高了 0.0022，测试集的 F1 分数提高了 0.0042。这表明 Transformer 可以学习到 GNN 模块难以学习的附加信息。

通过预训练和解冻的 GNN 模块，我们的 GraphTrans 可以获得更高的 F1 分数。这可能是因为 GNN 模块现在可以专注于学习局部结构信息，将远程信息学习留给其后的 Transformer 层。该模型受益于 [34]。请注意，对于表中的所有实验4，为了简单起见，我们没有将输入图的嵌入连接到 Transformer 的输入。

表 5：我们的 Graph Transformer 的 <CLS> 标记和特征连接的消融研究。这 mean 和 last 是序列分类中两种常用的嵌入聚合方法。

模型	有效的	测试
GraphTrans，均值	0.1398	0.1509
GraphTrans，最后	0.1566	0.1716
GraphTrans, <CLS>	0.1593	0.1784
GraphTrans，<CLS>，猫	**0.1670**	**0.1810**

表 6：我们的 GraphTrans 与 GCN-Virtual 在大型随机图上的训练迭代时间。我们保持模型设置与 OGBG-Code2 数据集上的相同。由于 GNN 训练的高成本邻居采样性质，我们的 GraphTrans 具有与 GCN 模型相似的可扩展性。

节点数	模型	边缘密度 20%	40%	60%	80%
500	GCN-虚拟 [18]	**44.3**	58.5	79.3	99.0
	GraphTrans (GCN)	48.4	**57.5**	**76.4**	**93.7**
1000	GCN-虚拟 [18]	99.1	171.8	249.5	断续器
	GraphTrans (GCN)	**96.9**	**168.4**	**244.3**	断续器
[illegible]	GCN-虚拟 [18]	131.8	237.7	断续	断续

5.5 <CLS> 嵌入的有效性

如图2b，我们可以观察到第 18 行（最后一行是 <CLS>）在多个列上呈现暗红色，这表明 <CLS> 学习关注图中的重要节点以学习整个图的表示。

我们还定量检查了 <CLS> 嵌入的有效性。在表中5，我们测试了几种常用的序列分类方法。平均操作将转换器的输出嵌入平均为单个图嵌入；最后一个操作将输出序列中的最后一个嵌入作为图嵌入。定量结果表明 <CLS> 嵌入是最有效的，在测试集上改进了 0.0275，因为模型可以学习从不同节点检索信息并将它们聚合到一个嵌入中。输入图中嵌入的连接和转换器的输入嵌入可以进一步提高验证和测试的 F1-score 到 0.1670 和 0.1733。

5.6 可扩展性

为了定量地衡量 GraphTrans 如何使用超过 100 个节点的大型图进行扩展，我们运行了一个迭代时间的微基准测试，用于在不同的图大小和边密度下进行训练。我们在具有不同数量的节点和边密度的随机生成的 Erdos-Renyi 图上训练基线。如表所示6，当节点数量和边密度增加时，我们的 GraphTrans 模型的扩展性至少与 GCN 模型一样好。 GCN 和 GraphTrans 都可以看到大型密集图的内存不足错误 (OOM)，但我们注意到 GraphTrans 的内存消耗与 GCN 基线相似。

5.7 计算效率

为了评估我们的 GraphTrans 在特定 GNN 主干上增加的开销，我们评估每次迭代的前向传递运行时间和后向传递运行时间。我们将模型归一化以具有大致相似的参数计数。结果如表所示7. 对于 NCI1 数据集，GraphTrans 实际上比类似的 GCN 模型训练得更快。对于 OGB-molpcba 和 OGB-Code2 数据集，GraphTrans 比基线 GNN 架构慢 7-11%。

表 7：我们的 GraphTrans 与 NCI1、OGBG-Molpcba 和 OGBG-Code2 数据集上的主干模块的前向和后向时间比较。 Speedup 是与基于 GNN 的模型相比的训练迭代速度；数字越大表示运行速度越快。

数据集	方法	转发时间（毫秒）	后退时间（毫秒）	加速
NCI1	GCN-虚拟 [18]	22.27 ± 2.04	14.35 ± 2.46	1.00×
	变压器	12.31 ± 1.68	9.32 ± 1.68	1.69×
	图形传输	15.01 ± 1.63	12.04 ± 2.25	1.35×
莫尔普巴	GCN-虚拟 [18]	14.79 ± 2.54	12.75 ± 3.00	1.00×
	变压器	12.34 ± 1.43	10.52 ± 1.60	1.20×
	图形传输	16.55 ± 2.93	14.3 ± 3.15	0.89×
代码2	GCN-虚拟 [18]	22.97 ± 6.13	38.53 ± 6.92	1.00×
	变压器	31.01 ± 9.00	33.30 ± 16.09	0.96×
	图形传输	34.93 ± 6.85	31.14 ± 12.90	0.93×

表 8：我们的 GraphTrans 的参数数量与基于 GNN 的模型在不同数据集上的比较。总体而言，GraphTrans 通过略微增加参数实现了更高的准确性。

数据集	神经网络	图形传输	三角洲
莫尔普巴	3.4M	4.2M	0.8M
新航	0.4M	0.5M	0.1M
代码2	12.5M	9.1M	-3.4M

5.8 参数数量

我们在表中比较了 GNN 基线和 GraphTrans 在不同数据集上的参数数量8.总体而言，GraphTrans 仅略微增加了 Molpcba 和 NCI 的总参数。对于 Code2，GraphTrans 比 GNN 的参数效率要高得多，同时将测试 F1 分数从 0.1629 提高到 0.1810。提高参数效率的一个原因是 Transformer 在昂贵的最终预测层之前减少了特征维度。

6 结论

我们提出了 GraphTrans，这是一个简单而强大的框架，用于学习与 GNN 的远程关系。利用最近的结果，这些结果表明结构先验对于高级别的长期关系可能是不必要的，甚至适得其反，我们使用随后的置换不变 Transformer 模块来增强标准 GNN 层堆栈。 Transformer 模块充当新的 GNN“读出”模块，同时允许学习图节点之间的成对交互并将它们总结为特殊的令牌嵌入，就像在 Transformers 的常见 NLP 应用程序中所做的那样。这个简单的框架在跨程序分析、分子和蛋白质关联网络的几个图分类任务中对现有技术进行了惊人的改进。在某些情况下，GraphTrans 优于尝试对特定于域的结构信息进行编码的方法。全面的，GraphTrans 提供了一种简单而通用的方法来改进远程图分类； 下一个方向包括节点和边缘分类任务的应用，以及 Transformer 对大图的进一步可扩展性改进。

7 致谢

我们感谢 Ethan Mehta、Azade Nazi、Daniel Rothschild、Adnan Sherif 和 Justin Wong 的深思熟虑的讨论和反馈。除了 NSF CISE 远征奖 CCF-1730628 之外，这项研究还得到了亚马逊网络服务、蚂蚁集团、爱立信、Facebook、Futurewei、谷歌、英特尔、微软、丰业银行和 VMware 的礼物的支持。