题目

摘要

为了捕捉更高阶的结构特征，大多数基于图神经网络（GNN）的算法会学习节点表示，并融入 k-hop 邻居的信息。由于查询 k-hop 邻居的时间复杂度较高，许多图算法无法在巨大密集的时间网络中进行毫秒级推理，这严重限制了图算法在某些领域，特别是金融欺诈检测中的应用潜力。因此，我们提出了一种名为“异步传播注意力网络”的异步连续时间动态图算法，用于实时时间图嵌入。传统的图模型通常执行两个串行操作：首先进行图查询，然后进行模型推断。与以往的图算法不同，我们将模型推断和图计算解耦，以减轻图查询操作对模型推断速度的影响。广泛的实验证明，我们提出的方法在显著提高推断速度的同时，能够达到竞争性能水平。

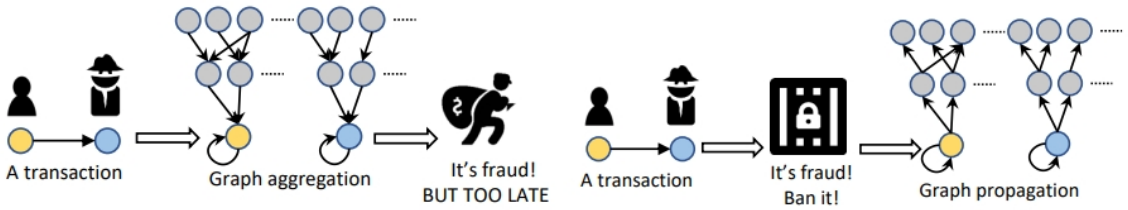
关键词：图神经网络；动态图；网络嵌入

1 引言

最近，基于 CTDG 的图算法在图挖掘社区中越来越受到关注。这种算法范式旨在处理节点之间的一批时间交互，相较于基于快照的 DTDG 算法，这将获得更大的弹性。大多数 CTDG 算法（如 TGAT 和 TGN）通过事件触发的时间子图聚合来建模动态节点状态。随着动态图嵌入学习的卓越性能，CTDG 算法在在线推理过程中面临着高延迟的问题。这些模型通常执行两个连续的操作：图查询和模型推理。当一批交互到来时，CTDG 算法需要首先访问它们的 k-hop 时间邻居。然后，模型将聚合这些邻居的信息以生成节点的嵌入。在实时时间网络中，CTDG 算法需要在线部署并完成实时推理，该过程应该在毫秒级别进行。

据我们所知，尚未有努力克服 CTDG 算法的这些严重问题。在本工作中，我们重新设计了 GNN 框架，将模型推理和图查询步骤解耦，以使繁重的图查询操作不会损害模型推理的速度。为了方便解释，我们将类似 TGAT 的算法称为同步 CTDG，将我们的 APAN 称为异步 CTDG。将繁重的查询和计算操作放入异步链接中可以将复杂的算法与在线业务决策系统隔离开来，从而获得更高的系统稳定性和可扩展性。

直观地说，异步 CTDG 可以满足我们的要求，但设计一个异步 CTDG 绝非像调整图计算阶段的顺序那样简单。异步传播注意力网络（APAN）是我们首次提出的符合上述异步 CTDG 算法框架的模型。APAN 有两个链接：同步推理链接和异步传播链接。在异步链接中，一旦交互完成，交互的详细信息将作为“邮件”传递到其 k-hop 邻居的“邮箱”中；在同步链接中，当交互发生时，APAN 不需要查询时间图中的邻居，而只需读取相关节点的“邮箱”并生成实时推理结果。



(a) Offline-deployed synchronous CTDG algorithm like TGAT [29]　(b) Online-deployed asynchronous CTDG algorithm APAN

图 1. (a) 由于用户无法容忍在庞大的图数据库中进行邻居查询所带来的高延迟，将同步的 CTDG 模型部署在在线支付平台几乎是毫无价值的。如果我们将其部署在离线系统中，可能无法在欺诈分子提取非法资金之前封禁账户，从而给我们的平台带来经济和声誉损失。(b) 异步传播注意网络（APAN）从根本上重新设计了 CTDG 算法的工作流程。图查询和计算阶段被转移到模型推断的后方；TGAT 利用图聚合技术来建模时态图结构，而 APAN 则使用图传播。APAN 满足我们在线部署的实时要求。

2 相关工作

此部分对课题内容相关的工作进行简要的分类概括与描述，二级标题中的内容为示意，可按照行文内容进行增删与更改，若二级标题无法对描述内容进行概括，可自行增加三级标题，后面内容同样如此，引文的 bib 文件统一粘贴到。

2.1 网络嵌入

网络嵌入 [2] 方法在保持网络拓扑属性和语义信息的同时，在学习节点、边、子图和整个图的低维表示方面取得了优异的性能。将静态图的节点嵌入到低维向量空间的方法层出不穷。DeepWalk [12] 首先生成随机作品对多重结构信息进行采样，并将节点序列放入由 Word2Vec 导出的 skip-gram 模型 [10] 中。Node2Vec [5] 平衡了宽度优先抽样和深度优先抽样，得到同态和结构等价的性质。此外，LINE [14] 和 SDNE [19] 不采用随机游走策略，而是基于 1 阶甚至更高阶节点相似度构建相似矩阵。也可以通过非负矩阵分解从邻接矩阵的拉普拉斯算子中导出表示。然而，这些图嵌入方法并不直接处理图结构，而是通过对节点和邻域的序列化，将其转换成类似于文本的线性结构。此外，基于随机行走的网络嵌入方法由于其换能性设置，难以处理不可见节点。

2.2 图神经网络

图卷积网络 [7](graph convolutional network, GCN) 作为 GNN 的主体，利用基于谱方法的卷积聚合器来收集信息，SAGE [6] 是将 GCN 扩展为归纳学习的综合改进，同样适用于图中的未知节点。此外，图注意网络 [18](GAT) 引入了一种注意机制来为每个邻居分配注意重要性。此外，一些尝试将 GNN 引入到编码器-解码器神经网络框架中。图自动编码器 (GAE) 和变分 GAE (VGAE) [8] 使用 GCN 构造编码器和解码器，从而可以从图结构信息和节点属性中以无监督的方式提取节点嵌入。此外，越来越多的研究人员尝试在图神经网络层中加入更多的跳跃连接，使每个节点从 k 跳之外的邻居那里聚合更多的信息，这可以通过非局部神经网络 (NLNN) 捕获远程依赖关系 [20]。由于已知邻域聚合方案的局限性，因此提出了 Jump Knowledge Network [22] 来学习自适应的、结构感知的表示。

2.3 离散时间动态图学习

由于网络演化和动态的复杂性，目前关于时态网络表征学习的研究较少。它们中的大多数都集中在学习几个快照中的节点表示，这些快照的边缘聚合在一个预定义的窗口中，这些快照的时间戳位于一个预定义的窗口中。这种类型的方法也称为 DTDG 算法。为了开发保留进化模式的可解释模型，DynamicTriad [24] 研究了三合一闭合过程，以捕捉开放三合一如何演变成封闭三合一;SPNN [1] 通过高阶依赖关系关注子图的架构，特别是将子图诱导为输入特征和模型架构;EPNE [15] 将周期和非周期模式分为两个通道，然后结合结构和时间信息。还有一些研究，如 E-LSTM-D [3] 和 Know-Evolve [16]，在学习表征时使用 LSTM 来保存历史信息和当前信息。

2.4 连续时间动态图学习

此外，在连续时间动态网络上，CTDNE [11] 建议在静态跳图模型中加入时间随机行走;HTNE [25] 将小贩过程应用于邻域形成;DynRep [17] 共同学习节点之间的拓扑演化和活动，其中节点 v 的表示在涉及 v 的事件发生后被更新。JODIE [9] 采用 RNN 模型更新相关节点的节点内存，而 TigeCMN [23] 采用键值存储网络 [4] 更新内存。同样，这两项工作都引入了一个注意力架构来读取节点内存并生成节点嵌入向量。然而，JODIE 和 TigeCMN 都没有显式地学习图的拓扑结构，因为它们只更新边缘上相关的两个节点，这意味着它们不能直接访问 2 跳邻居。TGAT [21] 利用时间嵌入核来定制 GAT 模型，获得时序图编码的能力。TGN [13] 结合了 JODIE 和 TGAT 的优点，在 TGAT 的时间聚合阶段引入了节点式内存。

3 本文方法

3.1 总体框架

图 2 提供了我们提出的 APAN 的概述，这是首个用于实时时态图嵌入的异步 CTDG 算法。APAN 的框架由基于注意力的编码器、基于多层感知机（MLP）的解码器和异步邮件传播模块组成。

同步部分：当发生交互时，编码器根据事件的详细信息、历史嵌入和邮箱数据更新节点嵌入。需要注意的是，如果一个节点在一批次中参与了多次交互，嵌入将仅生成一次。尽管如此，在整体上，所有参与新事件的节点的时态嵌入需要实时更新，这有助于模型捕捉图的动态变化。随后，MLP 解码器将利用这些更新的节点嵌入执行下游任务，如链路预测、节点分类、边缘分类和节点聚类。由于编码器和解码器都是前馈神经网络，我们无需在图数据库中查询图邻居，完成这两个阶段将非常迅速。

异步部分：在生成时态嵌入后，邮件传播器首先创建一封邮件，然后沿着时态边将其传播到 k-hop 邻居的邮箱。该邮件涵盖与发送节点相关的交互。由于邮件传播器位于异步链路上且不会影响用户体验，因此我们可以在此模块中进行一些更复杂的计算，比如从多个层次聚合邻居或计算子图统计值。

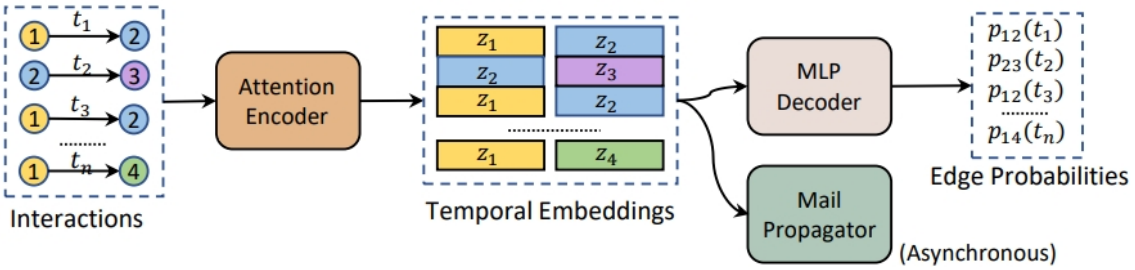


图 2. APAN 主要分为三个部分：编码器（Encoder）、解码器（Decoder）和传播器（Propagator）。请注意，编码器和解码器位于同步链路中，它们无需从图数据库中查询邻居的信息。因此，从交互发生到模型推断的时间延迟将非常短，用户将获得非常流畅的体验。在模型推断之后，邮件传播器将根据交互生成一封邮件，然后沿着时态边将其传播到 k-hop 邻居的邮箱。

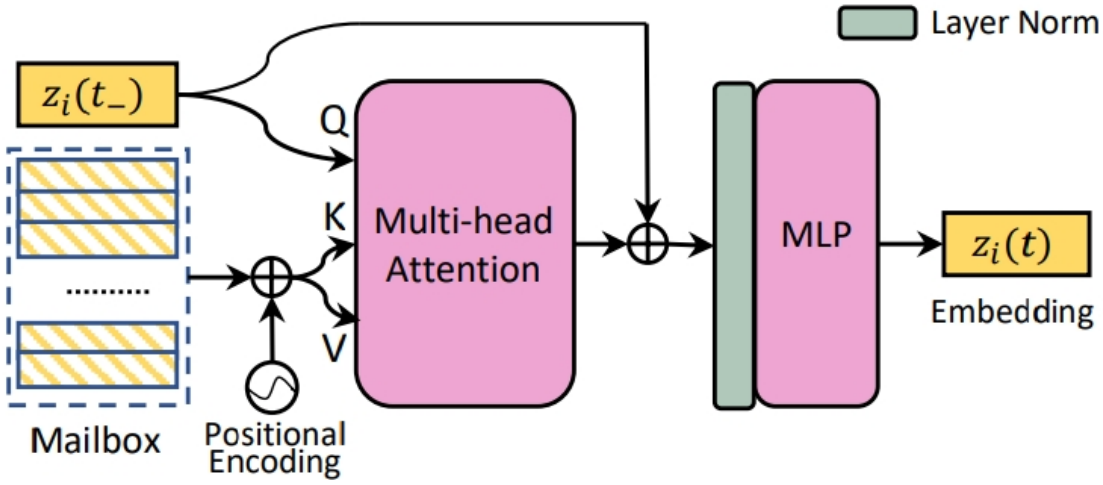


图 3. APAN 的编码器网络是一个多头注意力模块。这个注意力模块将根据上次更新的嵌入与邮箱中的邮件之间的关联性来计算当前节点嵌入。

3.2 基于注意力机制的编码器

图 3 详细展示了 APAN 中基于注意力的编码器的细节。该编码器引入了一个经典的注意力架构，以创建包含上一个嵌入和邮箱的当前节点嵌入。$z(t-)$ 表示节点在上一次参与交互时的状态。邮箱记录了邻居涉及的过去交互的详细信息，包括 k-hop 邻居。通过这种方式，编码器间接实现了时态邻居信息的聚合，以更新其节点嵌入。为实现这一目标，编码器由三个主要部分设计，它们是位置编码、多头注意力和层归一化。

位置编码：考虑到接收邮件的到达顺序，我们需要为每封单独的邮件尝试位置编码。由于我们已经设置了邮箱中邮件的最大数量，我们可以将位置信息转换为独热编码格式，然后将其馈送到嵌入查找层。嵌入查找层的输出是表示其静态特性的密集向量，更适合神经网络学习。

多头注意力：我们的编码器使用缩放的点积注意力作为注意力模块。注意力层的隐藏机制可以定义为：

$$\begin{aligned}\mathrm{Attn}(Q,K,V) &= \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V,\\ Q &= z(t-)W_Q,\\ K &= M(\hat{t})W_K,\\ V &= M(\hat{t})W_V.\end{aligned}$$

在实践中，注意力模型通常采用多个注意力头来形成多个子空间，并强制模型学习不同的信息方面。我们构建多个注意力并连接它们，以四个注意力头为例：

$$\begin{aligned}\mathrm{head}_i &= \mathrm{Attn}(Q_i,K_i,V_i), \quad i=1,\ldots,4,\\ \mathrm{MultiHead}(Q,K,V) &= \mathrm{Concat}(\mathrm{head}_1,\ldots,\mathrm{head}_4)W_O,\end{aligned}$$

层规范化：由于不同节点的注意力输出各不相同，我们需要一种规范化方案来限制输出的均值和方差。在注意力模型中，层规范化是最常见的选择，因为复杂的注意力机制可能破坏批处理中的统计分布。如果我们选择使用批次规范化，可能会导致次优的结果。通过计算用于规范化的均值和方差，这些均值和方差是来自层中所有神经元的输入的总和：

$$\begin{aligned}a &= \mathrm{MultiHead}(Q,K,V) + z(t-),\\ \mu &= \frac{1}{d}\sum_{i=1}^{d} a_i,\\ \sigma &= \sqrt{\frac{1}{d}\sum_{i=1}^{d}(a_i-\mu)^2},\\ \bar{a} &= f\left(\frac{g}{\sqrt{\sigma^2}}\odot(a-\mu)+b\right),\end{aligned}$$

其中 d 表示层规范化输入的维度，μ 和 σ 表示均值和方差，它们由层中所有隐藏单元共享。$\odot$ 是两个向量之间的逐元素乘法。可学习参数 b 和 g 分别定义为偏置和增益，以确保规范化操作对原始信息没有影响。

之后，层规范化的输出将传递到一个 MLP 网络，生成节点的新时态嵌入。

3.3 MLP 解码器

APAN 可以广泛应用于各种下游任务。注意力编码器和邮件传播器可以直接使用，无需任何架构更改，而 MLP 解码器需要进行微调以适应不同的任务。MLP 解码器的任务是利用时态节点嵌入为下游任务生成边预测。例如，如果我们需要预测两个节点之间是否会发生交互，那么这两个时态嵌入应该被串联为 $(z_i(t)||z_j(t))$ 并传递给解码器；如果我们需要判断一条边是否是欺诈交易，则这两个时态嵌入和边特征应该被串联为 $(z_i(t)||e_{ij}(t)||z_j(t))$。

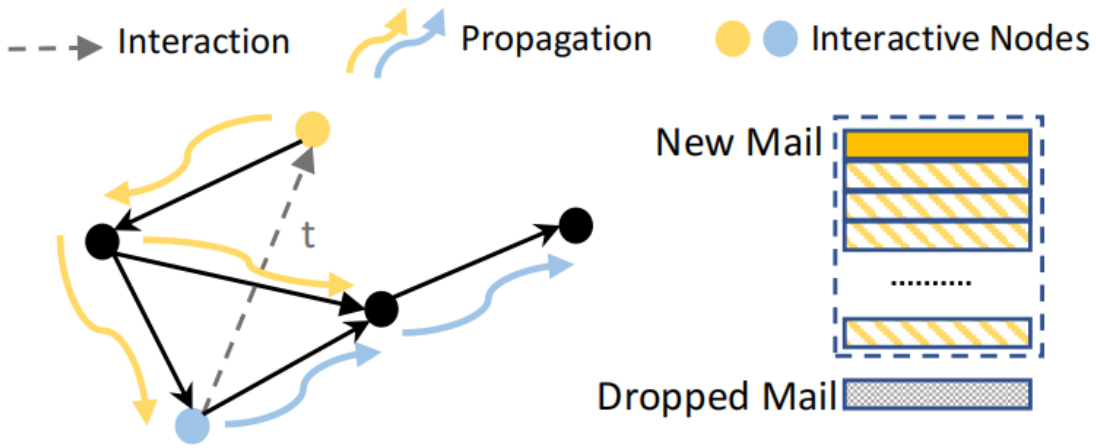


图 4. APAN 的编码器网络是一个多头注意力模块。这个注意力模块将根据上次更新的嵌入与邮箱中的邮件之间的关联性来计算当前节点嵌入。

3.4 异步邮件传播器

在图 4 中，我们以最简单的方式展示了我们的邮件传播器。黄色节点和蓝色节点在时间 t 时具有边特征 $e_{ij}(t)$ 的相互作用，然后这个相互作用可以被描述为一个元组 $(z_i(t), e_{ij}(t), z_j(t))$，其中 $(z_i(t)$ 和 $z_j(t))$ 是由注意力编码器创建的。APAN 中邮件传播的过程可以用以下两个数学公式描述：

$$\text{Mail:} \quad \text{mail}(t) = \phi(z_i(t), e_{ij}(t), z_j(t)),$$

$$\text{Mailbox:} \quad M(t) = \psi(M(t-), \rho(\{f(\text{mail}(t))\}_{N_{ij}^k})).$$

其中 $\phi(\cdot)$ 是生成邮件的函数，用于总结交互细节。N_{ij} 表示在互动节点 i 和 j 上引发的子图。$f(\cdot)$ 是邮件传递函数，定义了消息在传播中的衰减方式。$\rho(\cdot)$ 是降维函数，将多个传入的邮件聚合成一个单一的邮件。ψ 是更新函数，根据邮件更新邮箱。

请注意，邮件也标有时间戳。我们之所以使用求和而不是串联，是因为求和可以节省邮箱占用的内存容量。缺点是求和限制了节点嵌入的维度，可能在某些情况下获得次优解。

一旦生成了邮件，我们应该将其传递给其他节点，以便让它们知道它们的邻居发生了什么。然而，将某人的邮件传递给所有邻居是低效的。在大多数 GNN 算法中，节点消息是在采样的子图上传播的。我们采用最近邻居采样策略，因为 CTDG 方法旨在建模快速变化的趋势并更新节点嵌入。因此，最近邻居采样更容易恢复时变信息。

确定传播边界后，即邻居采样后，我们需要一个用于找到邮件的合理衰减或映射模式的函数。在 APAN 中，邮件传递函数简单地设置为一个识别函数。

在实践中，一个节点通常在邮件传播过程中收到多个邮件。活跃（高度）节点通常收到的邮件比不活跃（低度）节点多。为了避免这种不平衡，我们使用“均值”操作的缩减函数将多个邮件转换为一个。这样，每个节点在每个单一批次中只会收到一个邮件。这个机制确保了设计下一个邮箱更新模块的简便性。

一旦节点收到一封邮件，其邮箱应更新以总结节点邻居的历史状态。为了尽可能简洁，我们采用先进先出（FIFO）队列数据结构来更新邮箱。通过这个队列结构，邮箱将保留最新的信息并丢弃旧邮件。

3.5 异步 CTDG 框架

概念上，我们首次提出的异步 CTDG 框架旨在解决基于 GNN 的方法在毫秒级在线平台上难以部署的问题。APAN 只是符合该框架的最简单模型之一。在注意力编码器和邮件传播器中的几乎每个模块都有很大的改进空间。与我们提出的这些简单模块相比，其他更为复杂的模块可能具有更大的潜力来改进异步 CTDG 框架：

邮箱机制：我们介绍了引入邮箱的额外好处。邮箱机制对于在线部署的时态 GNN 模型是必要的。在流系统中（尤其是分布式流系统），我们无法保证事件将按照时间戳的顺序到达。因此，对于依赖于 RNN 动态更新的一些机器学习模型，例如 TGN 和 JODIE，这将给它们带来不稳定性。邮箱机制解决了这个严重的问题，因为存储在邮箱中的事件和消息在读取时将按照时间戳进行排序。

邮箱更新：键值内存网络框架为增强邮箱更新模块提供了一种可能的方向。

位置编码：可以利用 Xu 等人提出的时间嵌入核来编码时间戳，以取代我们 APAN 中的位置编码模块。

可解释性：假设在时间 t 发生了一次交互，一封邮件存储了详细信息，包括节点嵌入 $z_i(t)$,$z_j(t)$ 和边特征 $e_{ij}(t)$。然后，我们可以使用注意力权重来计算哪封邮件对最终节点嵌入产生了最大的影响。这种可解释性是其他模型所没有的，因为它们只存储了边特征 $e_{ij}(t)$，而没有存储 $z_i(t)$ 和 $z_j(t)$。然而，a) 引入太多的技巧可能会干扰读者对我们异步 CTDG 整体架构的信任。b) 我们确实完成了一些相关扩展的开发，但我们仅在实验数据集上进行了测试，并未将其部署在实际环境中。因此，我们将把这些改进放在未来的工作中。我们也期待来自图机器学习社区的研究人员提出更好的异步 CTDG 模型。

4 复现细节

4.1 与已有开源代码对比

本次复现的 APAN 模型在 github 有给出源码与 wikipedia 数据集介绍。因此复现过程参照操作文档。但是，该文档虚拟环境和依赖项版本指定不明确，最后经过试错整理出依赖关系，且给模型加入了日志功能来方便观察和记录结果。

4.2 实验环境搭建

采用的是 pytorch 的模型框架，主要的实验环境如下：

```
pytorch=1.5.0
DGL=0.5.2
numpy=1.19
python=3.8
```

4.3 创新点

本次实验的创新点主要是对现有工作的复现，并验证和分析结果。为代码完善了日志功能，并通过修改超参数，使得训练出的模型结果在推理性能上有些许提升。

以下是日志模块的 Python 代码：

```python
def set_logger():
    task_time = time.strftime("%Y-%m-%d_%H:%M", time.localtime())
    logging.basicConfig(
        level=logging.INFO,
        format ='%(asctime)s - %(levelname)s - %(message)s'
    )
    logger = logging.getLogger()
    logger.setLevel(logging.DEBUG)
    log_dir = Path("log/")

    log_file_path = log_dir / '{}.log'.format(task_time)
    fh = logging.FileHandler(log_file_path)
    fh.setLevel(logging.DEBUG)
    ch = logging.StreamHandler()
    ch.setLevel(logging.WARN)
    formatter = logging.Formatter(
        '%(asctime)s - %(levelname)s - %(message)s')
    fh.setFormatter(formatter)
    ch.setFormatter(formatter)
    logger.addHandler(fh)
    logger.addHandler(ch)
    return logger
```

5 实验结果分析

Batch size	Average Precision	AUC	Accuracy
128	96.96	97.08	91.08
200	98.16	98.21	93.88
256	98.06	98.16	92.98
512	97.94	98.06	93.25
800	96.83	98.84	91.01

表 1. batch size 对模型的影响

表 1 是 batch size 与模型性能的影响，整体看来，模型的健壮性非常好。我们可以发现

修改超参数 batch size，Average Precision、AUC、Accuracy 波动并不大。而 batch size 在一定程度上会影响模型训练与收敛的时间，同时也对 GPU 的显存有要求，APAN 的特性决定了它可以实用在不同大小的数据集上。

K-hop	Average Precision	AUC	Accuracy
6	98.42	98.64	94.57
7	98.73	97.88	93.88
8	98.06	98.34	93.98
9	98.51	96.84	95.12

表 2. K-hop 对模型的影响

表 2 是在训练过程中，最大选择几阶邻居的 mailbox 信息来进行嵌入。由表可知在下游分类任务中，讲 K-hop 取值在 6 的时候，AUC 和 Accuracy 分别能够达到 98.64 和 94.57。而且 APAN 采用的异步传播机制避免了同步模型中对最新交互信息的丢失，保持了最新交互信息的有效性，6 阶邻居信息为模型的推理效果提供了更为准确的特征嵌入。

Batch size	5-hop	6-hop	7-hop	8-hop
128	1.3471	1.0116	0.999	1.3658
200	1.2655	0.997	0.93434	1.20317
256	1.005	0.9987	0.9875	0.9634
512	1.768	1.2351	1.1146	1.21233

表 3. Batch size 与 K-hop 对模型推理速度的影响

表 3 是在下游任务中,模型推理速度与另外两个超参数的关系。整体观察不难发现,APAN 模型的推理速度非常快，batch size 选择在 256 左右能够在一定程度上提高模型处理下游任务的速度，例如分类。并且推理速度很快，能实现在互联网平台上实现更快速的推理，适用于诸如在线欺诈检测等对实时性要求较高的场景。

6 总结与展望

本论文引入了一种名为异步传播注意力网络（APAN）的新型算法框架，旨在解决实时时态图嵌入的问题。通过将传统的 CTDG 算法进行改进，APAN 能够适应在线部署和在互联网平台上进行实时推断。在广泛的实验中，我们验证了 APAN 在性能上的竞争力复现过程中的不足主要有两点：一是没有在更多的对及时性要求更高的数据集上进行验证；二是没有设计自己的下游任务发掘出 APAN 模型的应用潜力。

以下是一些潜在的研究方向值得深入探讨。首先，我们可以进一步优化 APAN 的性能，提高其在复杂网络结构和大规模数据集上的适应性。其次，可以考虑扩展 APAN 的应用范围，例如在其他领域中应用异步 CTDG 框架，以验证其通用性和可扩展性。此外，我们还可以探索结合深度学习和传统时态图嵌入方法的混合模型，以进一步提升性能和推广应用。

参考文献

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[3] Jinyin Chen, Jian Zhang, Xuanheng Xu, Chenbo Fu, Dan Zhang, Qingpeng Zhang, and Qi Xuan. E-lstm-d: A deep learning framework for dynamic network link prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(6):3699–3712, 2019.

[4] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 108–116, 2018.

[5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[7] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[8] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[9] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1269–1278, 2019.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[11] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion proceedings of the the web conference 2018*, pages 969–976, 2018.

[12] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

[13] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[14] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[15] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*, pages 3462–3471. PMLR, 2017.

[16] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*, pages 3462–3471. PMLR, 2017.

[17] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[19] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[20] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[21] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

[22] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018.

[23] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhao Li, and Can Wang. Learning temporal interaction graph embedding via coupled memory networks. In *Proceedings of the web conference 2020*, pages 3049–3055, 2020.

[24] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[25] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2857–2866, 2018.