GNN 预训练策略

姓名：杨艺

学号：2019214103

日期：2020 年 5 月 28 日

目录

图神经网络的预训练策略

1. 概述

本文解决的是图上的预训练问题。预训练解决的问题是：针对特定任务的标签数据有限，可以在有充足数据的相关任务上进行模型的预训练，然后再针对下游任务进行微调。本文提出一种新的 GNN 的预训练策略和自监督的方法[1]。本文方法的关键是在单个节点以及整个图上进行预训练，这样 GNN 就可以同时学习到局部和全局的表示信息。迁移学习指的是模型先在某些数据充足的任务上进行训练，然后在不同但相关的任务上重新使用。实验证明只在整张图上或者在单个节点上进行 GNN 的预训练，带来的提升很有限，甚至会在许多下游任务中带来负迁移。本文的策略避免了负迁移并且提升了多个下游任务的性能，在分子性质预测和蛋白质功能预测任务中超越了 state-of-the-art（目前最先进）。

2. 贡献

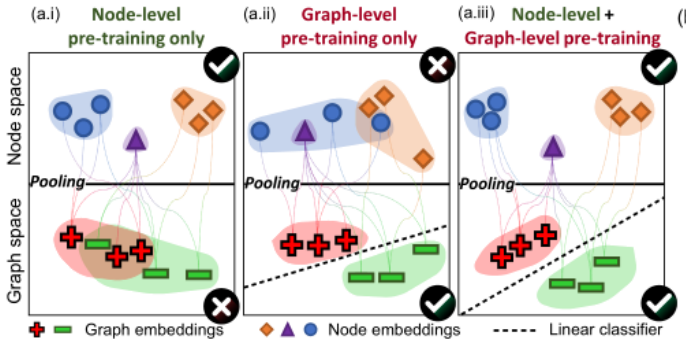
使用预训练方法实现 GNN 上的迁移学习，用于图级别的属性预测。贡献如下：

1) 提出第一个对 GNN 预训练的系统的大规模研究策略。建立了两个大的预训练数据集：有 2 百万张图的化学数据集，有 395K 张图的生物数据集。作者还表明了大规模的特定领域的数据集对研究预训练的重要性，而现有的下游基线数据集都太小了不足以评估模型的可靠性。
2) 提出了有效的 GNN 预训练策略，并且证明了其有效性以及在难迁移问题上对不均匀分布（out-of-distribution）数据的泛化能力。

3. 算法思想

作者提出了有效的 GNN 预训练策略，关键思想是：使用易获得的节点信息，让 GNN 捕获关于节点和边的特定领域的知识，以及图级别的知识。这使得 GNN 学习到了全局和局部层面的表示信息，如图 1 a.iii 所示。并使得 GNN 可以生成鲁

棒的可迁移到不同下游任务的图级别的表示（通过对节点表示的 pooling 得到），如图 1 所示。



(b) Categorization of our pre-training methods

	Node-level	Graph-level
Attribute prediction	Attribute Masking	Supervised Attribute Prediction
Structure prediction	Context Prediction	Structural Similarity Prediction

图 1 节点级+图预训练

注释：（a.i）仅使用节点级预训练时，可以很好地分离不同的节点，但是，节点嵌入是不可组合的，因此生成的图形嵌入是不可分离的。（a.ii）仅使用图级别的预训练，图嵌入被很好地分离，但是单个节点的嵌入不一定捕获其特定领域的语义。（a.iii）高质量的节点嵌入使得不同类型的节点可以很好地分离，而同时嵌入空间也是可组合的。这样就可以对整个图形进行准确而稳健的表示，并且可以将经过预训练的模型可靠地传输到各种下游任务。（b）对 GNN 的预训练方法进行分类。我们的方法（即上下文预测，属性屏蔽和图级别的有监督的预训练（监督的属性预测））都可以实现节点级和图的预训练。

4. 背景知识

4.1 图上的监督学习

定义图为 G=(V,E) ，节点属性为X_v，边属性为e_{uv}。给定图的集合{G1,...,GN}和标签{y1,...,yN}。图上的有监督学习的任务是学习到图的向量表示h_G为一个图预测标签：y_G=g(h_G)。例如，在分子特性预测中，G 是一个分子图，节点表示原子，边缘表示化学键，要预测的标签可以是毒性或酶结合。

4.2 什么是自监督学习

自监督学习主要是利用辅助任务（Pretext）从大规模的无监督数据中挖掘自身的监督信息，通过这种构造的监督信息对网络进行训练，从而可以学习到对下

游任务有价值的表征。（也就是说自监督学习的监督信息不是人工标注的，而是算法在大规模无监督数据中自动构造监督信息，来进行监督学习或训练。因此，大多数时候，我们称之为无监督预训练方法或无监督学习方法，严格上讲，他应该叫自监督学习）

自监督的 Pretrain - Finetune 流程：首先从大量的无标签数据中通过 pretext（为达到特定训练任务而设计的间接任务） 来训练网络（自动在数据中构造监督信息），得到预训练的模型，然后对于新的下游任务，和监督学习一样，迁移学习到的参数后微调即可。所以自监督学习的能力主要由下游任务的性能来体现。

4.3 邻域聚合 GNN

GNN 使用图上的连接已经节点和边的属性，为每个节点学习到向量表示 hv。GNN 通常是递归地聚合邻居信息和边的信息[2]，更新目标节点的信息。k 次迭代后，目标节点 v 的表示含有 k–hop 邻域的结构信息。第 k 层的 GNN 定义为：

$$h_v^{(k)} = COMBINE^{(k)}\left(h_v^{(k-1)}, AGGREGATE^{(k)}\left(\left\{\left(h_v^{(k-1)}, h_u^{(k-1)}, e_{uv}\right): u \in N(v)\right\}\right)\right)$$

其中$h_v^{(k)}$是节点 v 在第 k 次迭代/层的表示，e_{uv}是 u 和 v 之间的边的特征向量，$N(v)$是 v 的一组邻居，我们初始化$h_v^{(0)} = X_v$。

4.4 图表示学习

从最后一次迭代的输出中得到整张图的表示h_G：

$$h_G = READOUT\left(\left\{h_v^{(K)} \middle| v \in G\right\}\right)$$

READOUT 函数是一种不随输入排序变化而改变的函数，例如平均操作或图级别的池化函数[5]。

5. GNN 预训练的策略

本文训练策略的核心是在单个节点层面以及整张图层面进行 GNN 的预训练，这使得 GNN 能捕获两个层面的针对领域的语义信息。

5.1 节点级别的预训练

对于 GNNs 的节点级预培训，我们的方法是使用易得的无标签数据捕获图中特定领域的知识信息。作者提出两种自监督的方法：上下文预测(context prediction)和属性屏蔽(attribute masking)。

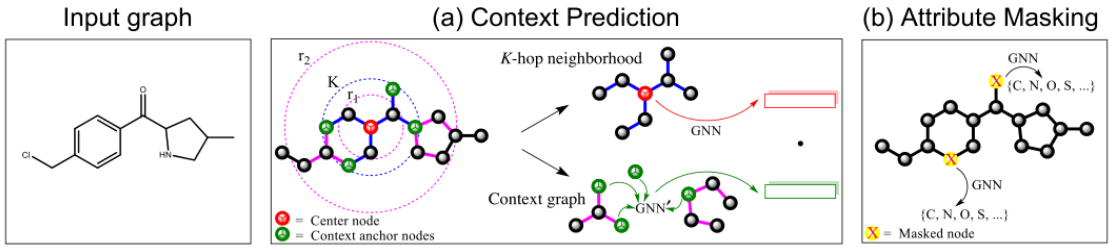


图 2 节点级方法

注释：（a）在上下文预测中，子图是围绕选定中心节点的 K 跳邻域，其中 K 是 GNN 层的数量，在图中设置为 2。上下文被定义为从中心节点开始的 r1 到 r2 跳之间的周围图形结构，在图中我们使用 r1=1 和 r2=4。（b）在属性屏蔽中，输入节点/边缘属性（如分子图中的原子类型）被随机屏蔽，并要求 GNN 预测它们。

5.1.1 上下文预测(context prediction)：使用图结构的分布

在上下文预测中，使用子图预测其周围的图结构。目标是训练出一个 GNN 可以将有着相似上下文结构的节点映射成相近的表示向量。

1) 邻居和上下文

图节点 v 的 K-hop 邻居包含和 v 距离不超过 K 的所有节点和边。节点 v 的上下文图定义为 v 邻居的图结构。用 K 层的 GNN 聚合 v 的 K 阶邻居信息，得到节点嵌入$h_v^{(K)}$。上下文图由 r1,r2 两个参数描述，表示 v 的 r1-hops 和 r2-hops 的子图（宽度为 r2-r1 的环）。

图 2 a 展示了邻居和上下文图。其中，r1<K，这样邻居和上下文图可以共享一部分节点，将这些节点看做上下文锚节点(context anchor nodes)。这些锚节点描述了邻居和上下文图是如何相互连接的。

2) 使用辅助的 GNN 将上下文编码为固定向量

由于图的组合特性，直接预测上下文图是很困难的。这不同于 NLP，文本中的词语都是来源于一个有限的词表。为了实现上下文的预测，作者使用辅助的 GNN（context GNN）将上下文图编码成固定长度的向量，如图 2 a 中的 GNN′ 所示。

如图 2 a 所示，应用上下文 GNN 得到上下文图的节点嵌入。然后对上下文锚节点的嵌入取平均，得到固定长度的上下文嵌入。节点 v 对应的上下文嵌入表示为c_u^G。

3) 使用负采样进行学习

使用负采样[4]联合学习 main GNN 和 context GNN。main GNN 将邻居编码获得节点嵌入，context GNN 将上下文图编码获得上下文的嵌入。

上下文预测的目标函数是一个二分类问题，判断特定的邻居和上下文图是否属于同一个节点：

$$\sigma\left(h_v^{(K)\top} c_{v'}^{G'}\right) \approx 1\{v \text{ and } v' \text{ are the same nodes}\}$$

- σ(·)表示 sigmoid 函数
- 1(·)是指示函数
- 正样本对：v′ = v, G′ = G
- 负样本对：从随机选择的图 G′ 中随机采样节点 v′
- 负样本采样率为 1，即正负样本数量相等，使用负对数似然函数作为损失函数。

预训练后的 main GNN 就是获得的预训练模型。

5.1.2 属性屏蔽(attribute masking)：使用图属性的分布

在属性屏蔽中，目的是通过学习图结构上节点/边属性分布的规律，捕获到领域知识。

属性遮掩的预训练工作如下：mask 掉节点/边的属性信息，基于邻域结构，使用 GNN 预测这些属性信息[5]。图 2 b 展示了这一方法在分子图中的应用。随机 mask 掉输入节点/边的属性，例如分子图中的原子类型，使用特殊的标志 mask 这些属性。然后使用 GNN 得到相应的节点/边的嵌入（边的嵌入可通过对边末尾

连接的节点嵌入求和得到）。最后应用线性模型预测 mask 的节点/边的属性。注意使用的是非全连接的图来捕获不同图结构中节点/边的分布规律。

这种 mask 节点和边属性的方法可用于有丰富注释的图，例如：

（1）分子图，节点的属性对应于原子的类型。捕获到这些属性在图中是如何分布的，有助于 GNN 学习到简单的化学规则（例如 化合价）以及更复杂的化学现象。

（2）在蛋白质互作网络(PPI)，边的属性对应于一对蛋白质中不同的互作关系。捕获这些属性在图中是如何分布的，有助于 GNN 学习到不同的交互是如何相互关联的。

5.2 图级别的预训练

有两种进行图级别的预训练方式：（1）对整张图的特定领域的属性做预测(例如 有监督的标签)；（2）对图结构做预测。

5.2.1 有监督的图级别的属性预测

由于图级别的表示h_G是直接用于下游预测任务的微调的，所以我们需要将特定领域的信息编码到h_G中。

通过定义有监督的图级别的预测任务，我们将图级别特定领域的知识编码到预训练的嵌入表示中。具体来说是使用图级别的多任务的有监督预训练，联合预测多个图的标签（这些标签有真实值）。例如，在蛋白质功能的预测中，目标是预测给定的蛋白质是否有给定的功能，可以预训练 GNN 来预测到目前为止已经被验证过的蛋白质的功能。为了联合预测多个图属性，每个属性都对应于一个二分类任务，在得到图的表示后经过一个线性分类器。

仅仅使用多任务图级别的预训练进行迁移学习的效果可能并不好，因为一些有监督的预训练任务可能和下游任务不相关，甚至会对下游任务的性能产生负作用(负迁移)。一种方法是选择相关的有监督的预训练任务，并且在在这些任务上预训练 GNN。但是这种方法通常不可取，因为选择出相关的任务需要专家的知识，而且预训练应该能应用于多个独立的不同的下游任务才对。为了解决这一问题，作者只使用多任务的有监督预训练进行图级别的学习，不使用在此过程中生

成的节点嵌入（如图 1 a.ii 所示）。这些无用的节点表示可能会加重负迁移问题，因为在节点的嵌入空间中，许多不同的预训练任务容易互相干扰。所以，本文预训练的策略是：首先进行节点级别的预训练，然后再进行图级别的预训练。这种方法可以生产更具有可迁移能力的图表示，并且有鲁棒性，可以提高下游任务的性能，不需要专家人为选择有监督的预训练任务。

5.2.2 结构相似度预测

第二种图级别预测任务方法的目的是：建模两个图间结构的相似性。这样的任务有：建模图的编辑距离[6]和预测图结构的相似性[7]。然而找到图距离的真实值是很困难的，大规模的图数据集中节点对数量巨大。这个方法超出了本文的范围，将其作为未来的工作。

5.3 概述：预训练 GNN 和下游任务微调

本文提出的预训练策略是：首先进行节点级别的自监督预训练，然后进行图级别的多任务有监督的预训练。这种方法可以生产更具有可迁移能力的图表示，并且有鲁棒性，可以提高下游任务的性能，不需要专家人为选择有监督的预训练任务。

GNN 的预训练结束后，将预训练得到的 GNN 模型在下游任务中进行微调。图级别的表示经过线性分类器后预测下游任务的图标签。

6. 相关工作

关于图中单个节点的无监督的方法学习图上的节点表示大致可分为两类：

（1）使用局部的基于随机游走的方法[8]，预测边是否存在，进而重构图的邻接矩阵，例如原始的边缘预测(EdgePred)[9]。

（2）Deep Graph Infomax 方法[10]，最大化局部节点表示和池化后的全局图的表示之间的互信息，训练得到节点编码器。

这些方法都是使得邻近的节点有相似的表示，在节点分类和链接预测任务中取得了很好的效果。但这种方法对于图级别的预测任务，可能不是最优的方法。

在图级别的预测任务中，捕获局部邻域的结构相似性通常比捕获图中节点的位置信息更重要。本文的方法同时考虑了节点级别和图级别的预训练任务，并且在实验中证明了，对预训练模型同时使用这两种类型的任务可以显著提高其性能。

也有一些工作对不同任务间节点嵌入的迁移进行了研究。但是提出的方法都是对不同的子结构使用不同的节点嵌入，没有进行参数共享。这种方法天然就是 transductive 的，不能实现不同数据集间的迁移，不能进行端到端的微调，由于数据的稀疏性也不能捕获到大量多样的邻居/上下文信息。

本文的 GNN 预训练方法解决了上述挑战，编码了图级别的和节点级别的依赖关系以及结构信息，并且可以共享参数。

7. 实验

实验任务：图分类（图级别的属性预测）。我们主要研究图同构网络 GIN[11]，这是用于图级预测任务的最具表现力和最先进的 GNN 架构。

7.1 数据集[1]

- 预训练数据集

1. 在化学领域从 ZINC15[12]数据集中采样了 2 million 个未标注的分子，用于节点级别自监督的预训练。使用 ChEMBL 数据集[13]进行图级别的多任务的有监督预训练。
2. 在生物领域使用从 PPI 网络中获得的 395K 个未标注的蛋白质 ego-networks 进行节点级别的自监督的预训练。使用 88K 个标注的蛋白质 ego-netowrks 进行图级别的多任务的有监督预训练，预测 5000 个 coarse-grained biological functions。

- 下游的用于分类任务的数据集

1. 在化学领域使用 MoleculeNet 中的 8 个二分类的数据集[14]。
2. 在生物领域从 PPI 网络[15]中获取数据。由 8 个不同物种的 88K 个蛋白质组成，其中以感兴趣的蛋白质为中心的子图（即 ego 网络）用于预测其生物功

[1] 数据集：http://snap.stanford.edu/gnn-pretrain

能。

7.2 代码实现

7.2.1 源码[2]项目结构

a. bio：生物学领域数据集预训练程序。
b. chem：化学领域数据集预训练程序。
c. model_architecture：存放 GCN,GAT,GraphSAGE 预训练模型的路径文件。
d. model_gin：存放 GIN 预训练模型的路径文件。
e. batch.py：批处理，将一批图建模为一个大的（双连通）图。
f. dataloader：从数据集中加载数据。
g. fintune.py：微调下游数据集使用的预训练 GNN 模型的参数。
h. loader.py：将 PPI 的网络图转换为 pytorch 几何数据对象。
i. model.py：模型定义，通过串联合并边缘信息扩展模型聚合。
j. pretrain_...：不同方法的预训练（分为自监督预训练和监督预训练）。
k. result_analysis.py：构造结果 dict，绘制不同预训练方法的训练和测试曲线。
l. splitter：对数据集进行分割。
m. util.py：导入数据集，数据输入。

[2] 源码：https://github.com/snap-stanford/pretrain-gnns/

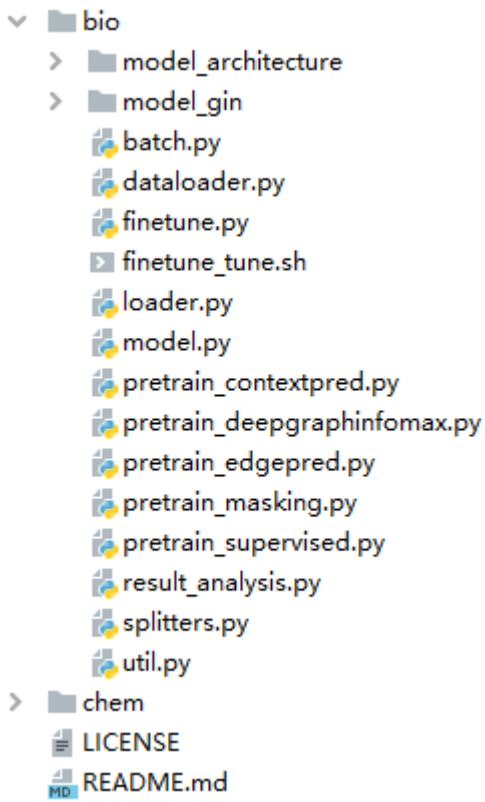


图 3 预训练项目结构

7.2.2 环境搭建与运行

1. 安装

我们使用以下 Python 软件包进行核心开发。已在 Python 3.7 上测试。

pytorch	1.0.1
torch-cluster	1.2.4
torch-geometric	1.0.3
torch-scatter	1.1.2
torch-sparse	0.2.4
torch-spline-conv	1.0.6
rdkit	2019.03.1.0

2. 数据集下载

对于化学数据集，请从化学数据（2.5GB）下载，解压缩，然后放在下 `chem/`。

对于生物学数据集，请从生物学数据（2GB）下载，解压缩并放在下 `bio/`。

3. 预训练和微调

a. 自监督预训练。将生成的预训练模型保存到 OUTPUT_MODEL_PATH。

python pretrain_contextpred.py

python pretrain_masking.py

python pretrain_edgepred.py

python pretrain_deepgraphinfomax.py

b. 监督预训练。加载预训练模型 INPUT_MODEL_PATH，使用监督预训练进一步对其进行预训练，然后将生成的预训练模型保存到 OUTPUT_MODEL_PATH。

python pretrain_supervised.py

c. 微调。使用下游数据集对预训练模型 INPUT_MODEL_PATH 进行微调，微调的结果将保存到 OUTPUT_FILE_PATH。

python finetune.py

4. 结果复现

我们可以通过运行 sh finetune_tune.sh SEED DEVICE 来复现文中的结果，其中 SEED 为 0 到 9 之间的随机种子，并 DEVICE 指定要运行脚本的 GPU ID。该脚本将在每个下游数据集上微调我们保存的预训练模型。

7.2.3 部分代码

以下代码以部分模型和方法为例。

- model.py：定义 GIN 模型，通过串联合并边缘信息扩展 GIN 聚合。参数：（1）emb_dim (int)：节点和边的嵌入维数。（2）input_layer (bool)：是否将 GINconv 应用于输入层（输入节点标签是统一的）。

代码：

```
class GINConv(MessagePassing):

    def __init__(self, emb_dim, aggr = "add", input_layer = False):
        super(GINConv, self).__init__()
        # 多层感知机
        self.mlp = torch.nn.Sequential(torch.nn.Linear(2*emb_dim, 2*emb_dim),
torch.nn.BatchNorm1d(2*emb_dim), torch.nn.ReLU(), torch.nn.Linear(2*emb_dim,
emb_dim))
```

```
        ### 将0/1边缘特征映射到嵌入
        self.edge_encoder = torch.nn.Linear(9, emb_dim)

        ###将统一的输入特征映射到嵌入.
        self.input_layer = input_layer
        if self.input_layer:
            self.input_node_embeddings = torch.nn.Embedding(2, emb_dim)

torch.nn.init.xavier_uniform_(self.input_node_embeddings.weight.data)

        self.aggr = aggr

    def forward(self, x, edge_index, edge_attr):
        #在边空间中添加自循环
        edge_index = add_self_loops(edge_index, num_nodes = x.size(0))

        #添加与自循环边相对应的特征
        self_loop_attr = torch.zeros(x.size(0), 9)
        self_loop_attr[:,7] = 1 # 自循环边的属性
        self_loop_attr =
self_loop_attr.to(edge_attr.device).to(edge_attr.dtype)
        edge_attr = torch.cat((edge_attr, self_loop_attr), dim = 0)

        edge_embeddings = self.edge_encoder(edge_attr)

        if self.input_layer:
            x = self.input_node_embeddings(x.to(torch.int64).view(-1,))

        return self.propagate(self.aggr, edge_index, x=x,
edge_attr=edge_embeddings)

    def message(self, x_j, edge_attr):
        return torch.cat([x_j, edge_attr], dim = 1)

    def update(self, aggr_out):
        return self.mlp(aggr_out)
```

- batch.py：将一批图建模为一个大的（双连通）图。torch_geometric.data.Data 类作为基类，它的所有方法可以在这里使用。另外，单图可以通过赋值向量进行重构。Batch 将每个节点映射到其各自的图形标识符。

  代码：

```python
class BatchFinetune(Data):

def __init__(self, batch=None, **kwargs):
    super(BatchMasking, self).__init__(**kwargs)
    self.batch = batch

@staticmethod
def from_data_list(data_list):
    r"""从 python 列表中构造一个批处理对象
    赋值向量:obj:`batch`是动态创建的."""
    keys = [set(data.keys) for data in data_list]
    keys = list(set.union(*keys))
    assert 'batch' not in keys

    batch = BatchMasking()

    for key in keys:
        batch[key] = []
    batch.batch = []

    cumsum_node = 0
    cumsum_edge = 0

    for i, data in enumerate(data_list):
        num_nodes = data.num_nodes
        batch.batch.append(torch.full((num_nodes, ), i, dtype=torch.long))
        for key in data.keys:
            item = data[key]
            if key in ['edge_index', 'center_node_idx']:
                item = item + cumsum_node
            batch[key].append(item)

        cumsum_node += num_nodes
        cumsum_edge += data.edge_index.shape[1]

    for key in keys:
        batch[key] = torch.cat(
            batch[key], dim=data_list[0].cat_dim(key, batch[key][0]))
    batch.batch = torch.cat(batch.batch, dim=-1)
    return batch.contiguous()

@property
def num_graphs(self):
```

```
        """返回批处理中的图形数"""
        return self.batch[-1].item() + 1
```

- dataloader.py：从 torch_geometric.data.dataset 类中小批量加载数据。参数：（1）dataset (Dataset)：加载数据的数据集。（2）batch_size（int, optional）：每个批次如何加载样本。（3）shuffle (bool, optional)：如果设置为 Ture，则数据将在每个 epoch 重洗。

  代码：

```
class DataLoaderFinetune(torch.utils.data.DataLoader):

    def __init__(self, dataset, batch_size=1, shuffle=True, **kwargs):
        super(DataLoaderFinetune, self).__init__(
            dataset,
            batch_size,
            shuffle,
            collate_fn=lambda data_list:
BatchFinetune.from_data_list(data_list),
            **kwargs)
```

7.3 实验结果

我们将我们的预训练策略与两种简单的基线策略进行了全面比较：（i）对相关图形级任务进行广泛的监督多任务预训练，以及（ii）节点级自我监督的预训练。

Dataset		BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	Average
# Molecules		2039	7831	8575	1427	1478	93087	41127	1513	/
# Binary prediction tasks		1	12	617	27	2	17	1	1	/
Pre-training strategy		Out-of-distribution prediction (scaffold split)								
Graph-level	Node-level									
–	–	65.8 ±4.5	74.0 ±0.8	63.4 ±0.6	57.3 ±1.6	58.0 ±4.4	71.8 ±2.5	75.3 ±1.9	70.1 ±5.4	67.0
–	Infomax	**68.8 ±0.8**	75.3 ±0.5	62.7 ±0.4	58.4 ±0.8	69.9 ±3.0	75.3 ±2.5	76.0 ±0.7	75.9 ±1.6	70.3
–	EdgePred	67.3 ±2.4	76.0 ±0.6	64.1 ±0.6	60.4 ±0.7	64.1 ±3.7	74.1 ±2.1	76.3 ±1.0	79.9 ±0.9	70.3
–	AttrMasking	64.3 ±2.8	76.7 ±0.4	64.2 ±0.5	61.0 ±0.7	71.8 ±4.1	74.7 ±1.4	77.2 ±1.1	79.3 ±1.6	71.1
–	ContextPred	68.0 ±2.0	75.7 ±0.7	63.9 ±0.6	60.9 ±0.6	65.9 ±3.8	75.8 ±1.7	77.3 ±1.0	79.6 ±1.2	70.9
Supervised	–	68.3 ±0.7	77.0 ±0.3	64.4 ±0.4	62.1 ±0.5	57.2 ±2.5	79.4 ±1.3	74.4 ±1.2	76.9 ±1.0	70.0
Supervised	Infomax	68.0 ±1.8	77.8 ±0.3	64.9 ±0.7	60.9 ±0.6	**71.2 ±2.8**	**81.3 ±1.4**	77.8 ±0.9	80.1 ±0.9	72.8
Supervised	EdgePred	66.6 ±2.2	**78.3 ±0.3**	**66.5 ±0.3**	**63.3 ±0.9**	70.9 ±4.6	78.5 ±2.4	77.5 ±0.8	79.1 ±3.7	72.6
Supervised	AttrMasking	66.5 ±2.5	77.9 ±0.4	65.1 ±0.3	**63.9 ±0.9**	**73.7 ±2.8**	**81.2 ±1.9**	77.1 ±1.2	80.3 ±0.9	73.2
Supervised	ContextPred	**68.7 ±1.3**	**78.1 ±0.6**	65.7 ±0.6	62.7 ±0.8	**72.6 ±1.5**	**81.3 ±2.1**	**79.9 ±0.7**	**84.5 ±0.7**	**74.2**

表 1 使用不同的 GIN 训练前策略测试 ROC-AUC 在分子预测基准上的表现

注释：最右边的列平均了 8 个数据集的测试性能平均值。注意，节点级和图级的预训练对于良好的性能是必不可少的。

	Chemistry			Biology		
	Non-pre-trained	Pre-trained	Gain	Non-pre-trained	Pre-trained	Gain
GIN	67.0	**74.2**	**+7.2**	64.8 ± 1.0	**74.2 ± 1.5**	**+9.4**
GCN	**68.9**	72.2	+3.4	63.2 ± 1.0	70.9 ± 1.7	+7.7
GraphSAGE	68.3	70.3	+2.0	65.7 ± 1.2	68.5 ± 1.5	+2.8
GAT	66.8	60.3	-6.5	**68.2 ± 1.1**	67.8 ± 3.6	-0.4

表 2 测试不同 GNN 架构在训练前和训练前的 ROC-AUC（百分比）表现

注释：在没有预先训练的情况下，表现力较弱的 gnn 比表现力最强的 GIN 提供稍好的性能，因为它们在低数据状态下的模型复杂度较小。然而，通过预训练，最具表现力的 GIN 被适当地规则化，并支配其他架构。化学数据的预训练策略：上下文预测+图形级监督预训练；生物数据的预训练策略：属性屏蔽+图形级监督预训练。

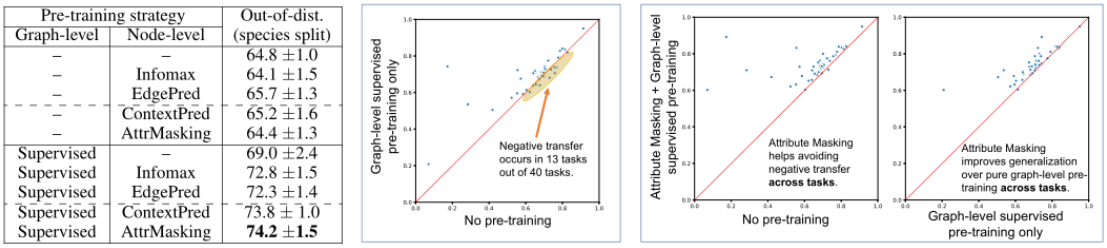


图 4 使用 GIN 的不同预训练策略测试蛋白质功能预测的 ROC-AUC

注释：（左）测试由不同训练前策略获得的 ROC-AUC 分数，其中分数在 40 个细粒度预测任务中平均。（中和右）：40 个个体下游任务的一对训练前策略的 ROC-AUC 得分的散点图比较。每个点代表一个特定的下游任务。（中间）：有许多下游任务，其中图级多任务监督预训练模型的性能比非预训练模型差，表明负迁移。（右）：当图级多任务监督预训练和属性掩蔽相结合时，避免了在下游任务产生的负迁移。与纯图级监督预训练相比，性能也有所提高。

我们在表 2 和 1 以及图 3 中报告分子特性预测和蛋白质功能预测的结果。我们的系统研究提出了以下趋势：

观察 1：表 2 显示了最具表现力的 GNN 架构（GIN），经过预训练后，可在整个域和数据集中实现最佳性能。与通过 GIN 架构获得的预训练收益相比，使用表达较低的 GNN（GCN，GraphSAGE 和 GAT）获得的预训练收益较小，有时甚至为负（表 2）。这一发现证实了以前的观察结果[16]，使用表达模型对于充分利

用预训练至关重要，并且在具有有限表达能力的模型上使用预训练甚至会损害性能，例如 GCN，GraphSAGE 和 GAT。

观察 2：从表 1 的阴影单元格和图 3 中间面板的突出显示区域可以看出，执行 GNN 的广泛图形级多任务监督预训练的强基线策略提供了令人惊讶的有限性能增益，并在许多下游任务上产生负迁移（分子预测中 8 个数据集中的 2 个，蛋白质功能预测的 40 项任务中有 13 项）。

观察 3：从表 1 的上半部分和图 3 的左面板，我们看到另一个基线策略，它只执行节点级的自监督预训练，也给出了有限的性能改进，与图级的多任务监督预训练基线相当。

观察 4：从表 1 的下半部分和图 3 的右面板可以看出，我们的图级多任务监督和节点级自监督预训练相结合的预训练策略避免了在下游数据集产生的负迁移，获得了最佳的性能。

观察 5：此外，从表 1 和图 3 的左面板可以看出，我们的策略提供了比两种基线预训练策略以及非预训练模型明显更好的预测性能，从而实现了最新的性能。

8.总结

我们开发了一种新的 GNNs 预训练策略。我们的策略成功的关键是结合一个表达性强的 GNN 来考虑节点级和图级的预训练。这确保了节点嵌入捕获到局部邻域语义信息，这些语义信息被汇集在一起以获得有意义的图级表示，而图级表示又被用于下游任务。在多个数据集、不同的下游任务和不同的 GNN 体系结构上的实验表明，这一预训练策略比没有经过预训练的模型具有更强的对 out-of-distribution 的泛化能力。

参考文献

[1] W. Hu*, B. Liu*, J. Gomes, M. Zitnik., P. Liang, V. Pande, J. Leskovec. Strategies for Pre-training Graph Neural Networks.International Conference on Learning Representations (ICLR), 2020.

[2] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In International Conference on Machine Learning (ICML), pp. 1273–1272, 2017.

[3] Rex Ying, Jiaxuan Y ou, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In Advances in Neural Information Processing Systems (NIPS), 2018.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems (NeurIPS), pp. 3111–3119, 2013.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2019.

[6] Y unsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. Unsupervised inductive whole-graph embedding by preserving graph proximity. In International Joint Conference on Artificial Intelligence (IJCAI), 2019.

[7] Nicolò Navarin, Dinh V Tran, and Alessandro Sperduti. Pre-training graph neural networks with kernels. arXiv preprint arXiv:1811.06930, 2018.

[8] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 855–864. ACM, 2016.

[9] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems (NeurIPS), pp. 1025–1035, 2017a.

[10] Petar V eli˘ ckovi´ c, William Fedus, William L Hamilton, Pietro Liò, Y oshua Bengio, and R Devon Hjelm. Deep graph infomax. In International Conference on Learning Representations (ICLR), 2019.

[11] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks,” in Proc. of ICLR, 2019.

[12] Teague Sterling and John J. Irwin. Zinc 15 – ligand discovery for everyone. Journal of Chemical Information and Modeling, 55(11):2324–2337, 2015. doi: 10.1021/acs.jcim.5b00559. PMID:26479676.

[13] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. Chemical Science, 9(24):5441–5451, 2018.

[14] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. Chemical science, 9(2):513–530, 2018.

[15] Marinka Zitnik, Rok Sosi˘ c, Marcus W. Feldman, and Jure Leskovec. Evolution of resilience in protein interactomes across the tree of life. Proceedings of the National Academy of Sciences, 116(10):4426–4433, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1818013116. URL https://www.pnas.org/content/116/10/4426.

[16] Dumitru Erhan, Y oshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research, 11(Feb):625–660, 2010.