

# 论文阅读笔记

郭治焱

zhiyanguo@hnu.edu.cn [www.gzyzq.site](http://www.gzyzq.site)

September 2, 2021

## Contents

|   |    |
|---|----|
| List of Figures   | 3  |
| 1 GraphRNN: Generating Realistic Graphs with Deep Auto-Regressive Models                            | 5  |
| 2 How powerful are graph gnns?  | 8  |
| 3 Gated Graph Sequence Neural Networks  | 10 |
| 4 Graph Structure of Neural Networks  | 12 |
| 5 Variational Graph Auto-Encoders   | 13 |
| 6 Hierarchical Graph Representation Learning with Differentiable Pooling                            | 14 |
| 7 Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation                   | 15 |
| 8 Auto-Encoding Variational Bayes   | 16 |
| 9 Deep Graph random Process for Relational-Thinking-based Speech Recognition                        | 18 |
| 10 Adversarial Autoencoders   | 21 |
| 11 A Comprehensive Survey on Graph Neural Network   | 23 |
| 12 The Emerging Field of Signal Processing on Graphs  | 27 |
| 13 Rethinking pooling in graph neural networks  | 28 |
| 14 FASTGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling                 | 29 |
| 15 Weakly-Supervised Disentanglement Without Compromises  | 29 |
| 16 Representation Learning: A Review and New Perspectives   | 30 |
| 17 Contrastive and Generative Graph Convolutional Networks for Graph-based Semi-Supervised Learning | 32 |

|  |           |
|--|-----------|
| <b>18 Inductive and Unsupervised Representation Learning on Graph Structured Objects</b>               | <b>34</b> |
| <b>19 Attention is All you need</b>  | <b>36</b> |
| <b>20 Deep Residual Learning for Image Recognition</b>   | <b>39</b> |
| <b>21 DeepGCNs: Can GCNs Go as Deep as CNNs?</b>   | <b>41</b> |
| <b>22 Combining Label Propagation and Simple Models OUT-PERFORMS Graph Networks</b>                    | <b>43</b> |
| <b>23 On the Bottleneck of Graph Neural Networks And Practical Implcation</b>                          | <b>45</b> |
| <b>24 Representation Learning via Invariant Causal Mechanism</b>                                       | <b>46</b> |
| <b>25 U-Net: Convolutional Networks for Biomedical Image Segmentation</b>                              | <b>47</b> |
| <b>26 UNSUPERVISED REPRESENTATION LEARNING FOR TIME SERIES WITH TEMPORAL NEIGHBORHOOD CODING</b>       | <b>48</b> |
| <b>27 Graph Similarity</b>   | <b>50</b> |
| 27.1 graph2vec: Learning Distributed Representation of Graphs . . . . .                                | 50        |
| 27.2 Convolutional Set Matching for Graph Similarity . . . . .   | 51        |
| 27.3 SimGNN: A Neural Network Approach to Fast Graph Similarity Computation . . . . .                  | 53        |
| 27.4 Unsupervised Inductive Graph-level Representation Learning via Grpah-Graph proximity . . . . .    | 55        |
| 27.5 Grapg-Graph Similarity Network . . . . .  | 56        |
| 27.6 Deep Graph Similarity Learning: A Survey . . . . .  | 57        |
| 27.7 Hierarchical Large-scale Graph Similarity Computation via Graph Coarsening and Matching . . . . . | 58        |
| 27.8 Deep Graph Kernels . . . . .  | 59        |
| <b>28 Dynamic Graphs</b>   | <b>61</b> |
| 28.1 EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs . . . . .                     | 61        |
| 28.2 Temporal Graph Networks for Deep Learning on Dynamic Graphs . . . . .                             | 62        |
| 28.3 Inductive representation learning on Temporal Graphs . . . . .                                    | 64        |
| 28.4 DySAT: Deep Neural Representation Learning on Dynamic Graph via Self-Attention Networks . . . . . | 65        |
| 28.5 Spatial Temporal Graph Convolutional Network for Skeleton-Based Action Recogonition . . . . .     | 67        |
| 28.6 Inductive Representation Learning In Temporal Networks via Causal Anonymous Walks . . . . .       | 68        |
| 28.7 DISCRETE GRAPH STRUCTURE LEARNING FOR FORECASTING MULTIPLE TIME SERIES . . . . .                  | 71        |
| <b>29 Recommender System</b>   | <b>72</b> |
| 29.1 Factorization Machines . . . . .  | 72        |
| 29.2 Wide & Deep Learning for Recommender Systems . . . . .  | 74        |
| 29.3 Deep Neural Networks for YouTube Recommendations . . . . .  | 75        |
| 29.4 Field-aware Factorization Machines for CTR Prediction . . . . .                                   | 77        |
| 29.5 DeepFM: A Factorization-Machine based Neural Network for CTR Prediction . . . . .                 | 79        |

|   |           |
|---|-----------|
| <b>30 简读论文</b>  | <b>82</b> |
| 30.1 Embedding Words in Non-Vector Space with Unsupervised Graph Learning . . . . .           | 82        |
| 30.2 Accurate, Efficient and Scalable Training of Graph Neural Networks . . . . .             | 82        |
| 30.3 Learning to Represent Image and Text with Denotation Graph . . . . .                     | 82        |
| 30.4 Towards Expressive Graph Representation . . . . .  | 83        |
| 30.5 Enhancing Extractive Text Summarization with Topic-Aware Graph Neural Networks . . . . . | 83        |
| 30.6 Editing Graphs to Satisfy Diversity Requirements . . . . .                               | 84        |
| 30.7 On the exact computation of the graph edit distance . . . . .                            | 84        |
| 30.8 Retrieval-Augmented Generation for Code Summarization via Hybrid GNN . . . . .           | 84        |
| <b>References</b>   | <b>85</b> |

## List of Figures

|   |    |
|---|----|
| 1 GraphRNN 生成图的过程 . . . . .   | 6  |
| 2 令 Mean, Max 失败的例子 . . . . .                                       | 8  |
| 3 LSTM 和 GRU cell 的内部结构 . . . . .                                   | 10 |
| 4 GG-NN . . . . .   | 11 |
| 5 GGS-NN . . . . .  | 11 |
| 6 VGAE . . . . .  | 13 |
| 7 Overview of DIFFPOOL . . . . .                                    | 14 |
| 8 Overview of GCPN . . . . .  | 15 |
| 9 有向图模型 . . . . .   | 16 |
| 10 without vs. with reparameterization . . . . .                    | 18 |
| 11 Architecture of RTN for acoustic modelling . . . . .             | 21 |
| 12 Architecture of AAE . . . . .                                    | 22 |
| 13 RecGNNs v.s. ConvGNNs . . . . .                                  | 24 |
| 14 Difference between GCN and GAT . . . . .                         | 25 |
| 15 Main characteristics of selected GAEs . . . . .                  | 25 |
| 16 Summary of benchmark datasets . . . . .                          | 27 |
| 17 CG3 . . . . .  | 33 |
| 18 SEED . . . . .   | 34 |
| 19 WEAVE 与随机游走对比 . . . . .  | 35 |
| 20 Transformer Architecture . . . . .                               | 37 |
| 21 Scaled Dot-Production Attention 与 Multi-Head Attention . . . . . | 38 |
| 22 CNN、Attention、Multi-head Attention . . . . .                     | 39 |
| 23 Degradation . . . . .  | 40 |
| 24 Residual learning: a building block . . . . .                    | 40 |
| 25 Traing deep GCNs . . . . .                                       | 41 |
| 26 Dilated Convolution in GCNs . . . . .                            | 42 |
| 27 DeepGCN . . . . .  | 42 |
| 28 Correct and Smooth . . . . .                                     | 44 |
| 29 The bottlenecks of Seq2seq and GNN models . . . . .              | 45 |
| 30 Data Generation in Causal interpretation . . . . .               | 46 |

|    |   |    |
|----|---|----|
| 31 | U-Net architecture . . . . .                                    | 48 |
| 32 | Overlap-tile strategy . . . . .                                 | 49 |
| 33 | TNC Architecture . . . . .                                      | 49 |
| 34 | GSimCNN 的三个阶段，分别用不同颜色的虚线框圈出 . . . . .                           | 51 |
| 35 | 左边的为相似的图的相似矩阵，右边为不相似图的相似矩阵 . . . . .                            | 52 |
| 36 | SimGNN . . . . .  | 53 |
| 37 | Overview of UGraphEmb . . . . .                                 | 55 |
| 38 | Overview of Graph-Graph . . . . .                               | 56 |
| 39 | Taxonomy of DGS . . . . .                                       | 57 |
| 40 | Overview of COSIM-GNN . . . . .                                 | 59 |
| 41 | EvolveGCN . . . . .   | 61 |
| 42 | TGN . . . . .   | 63 |
| 43 | TGAT layer，采用了多头注意力机制， $k=3$ . . . . .                          | 65 |
| 44 | DySAT . . . . .   | 66 |
| 45 | ST-GCN . . . . .  | 68 |
| 46 | Anonymous walks . . . . .                                       | 69 |
| 47 | CAW . . . . .   | 70 |
| 48 | Temporal Walk Extraction . . . . .                              | 70 |
| 49 | GTS Architecture . . . . .                                      | 71 |
| 50 | Wide&Deep . . . . .   | 74 |
| 51 | 推荐系统整体架构 . . . . .  | 75 |
| 52 | Deep candidate generation model architecture . . . . .          | 76 |
| 53 | Deep ranking network architecture . . . . .                     | 77 |
| 54 | Training FFM with SGD . . . . .                                 | 78 |
| 55 | Wide & deep architecture of DeepFM . . . . .                    | 80 |
| 56 | The architecture of FM . . . . .                                | 80 |
| 57 | The architecture of DNN . . . . .                               | 81 |
| 58 | The structure of the embedding layer . . . . .                  | 81 |
| 59 | Dense Embedding 层计算方式 . . . . .                                 | 81 |
| 60 | denotation graph extracted from the FLICKR30K dataset . . . . . | 83 |
| 61 | Overview of Topic-GraphSum . . . . .                            | 84 |
| 62 | Overview of HGNN . . . . .                                      | 85 |
| 63 | An Example of Code Property Graph(CPG) . . . . .                | 85 |

# 1 GraphRNN: Generating Realistic Graphs with Deep Auto-Regressive Models

论文地址: <https://arxiv.org/abs/1802.08773>

源码地址: [GraphRNN](#)

关键词: RNN, Graph generation

写于: 2020-09-18

论文 [68] 提出了一个深度自回归 (deep autoregressive) 模型, 用于图的生成。GraphRNN 使用序列的生成概念对图的生产进行建模。为了定量地评估 GraphRNN 的性能, 论文中使用 MMD(Maximum Mean Discrepancy) 的变体来衡量训练的图数据集和生成的图数据集之间的差异。

首先介绍图生成的应用。生产与真实图类似的图在很多领域都有重要的应用, 例如对物理、社会进行建模, 药物、分子结构的发现, 构建知识图谱等。现有的图生成的方法可以分为以下几种:

传统的图生成方法中, 有 Kronecker graphs, exponential random graphs, stochastic models 等。传统的方法主要依靠手工构建特征, 来生成指定特征的图, 这种方法并不能直接在图数据 (例如很多分子的图) 中学习模型进而得到图生成模型。

近期也有一些深度生成模型被提出, 如 VAE(variational autoencoders), GAN(generative adversarial networks)[24, 16]。并且已经有一些基于深度生成模型的图生成方法, 如 [53, 29]。但是这些深度生成模型在生成图时依然存在一些不足, 如只能从单一的图进行学习或者只能生成结点数较少的图 (如少于 40)。

图生成问题中的几个难点:

- 生成空间太大且可变。要制定具有  $n$  个结点的图需要  $n^2$  个值来确定边, 并且结点和边的数量是不确定的。
- 表示不唯一。个人理解是因为图的同构性导致的。
- 复杂的依赖关系。边的生成并不是独立的, 先后生成的边之间是存在依赖关系。[29] 中提出了解决这个问题的方法, 但是复杂度太高。

接下来是 GraphRNN 的细节。首先先看看如何将一个图看作序列生成的过程。对于一个给定的图  $G$ , 图中共  $n$  个结点, 下式中  $A_{ij}^\pi$  为  $G$  的邻接矩阵,  $\pi$  为结点的某个排列。则  $G$  可以表示成如下形式:

$$S^\pi = (S_1^\pi, \dots, S_n^\pi)$$

其中  $S_i^\pi = (A_{1,i}^\pi, \dots, A_{i-1,i}^\pi), \forall i \in 2, \dots, n$ , 其实就相当于邻接矩阵的第  $i$  列的上半部分, 表示第  $i$  个结点与之前  $i - 1$  个结点的连接关系。Okay, 图的序列表示解释到此为止。

根据以上的解释, 则一个图出现的概率  $p(G)$  可以表示为联合概率分布  $p(G, S^\pi)$  关于  $G$  的边缘分布, 令  $S$  表示  $G$  所有可能的排列, 即

$$p(G) = \sum_{S^\pi \in S} p(S^\pi)$$

对于某个排列  $S^\pi$ ,  $p(S^\pi) = \prod_{i=1}^{n+1} p(S_i^\pi | S_1^\pi, \dots, S_{i-1}^\pi)$ , 将  $p(S_i^\pi | S_1^\pi, \dots, S_{i-1}^\pi)$  简化表示为  $p(S_i^\pi | S_{<i}^\pi)$ 。其实意义很明显, 即当前结点的某条边的概率取决于该结点已经有的边情况。

现在假设已经获得了生成模型, 那么如何去生成一个图呢? 过程如下。

---

**Algorithm 1** GraphRNN inference algorithm

---

**Input:** RNN-based transition module  $f_{trans}$ , output module  $f_{out}$ , probability distribution  $\mathcal{P}_{\theta_i}$  parameterized by  $\theta_i$ , start token  $SOS$ , end token  $EOS$ , empty graph state  $h'$

**Output :** Graph sequence  $S^\pi S_1^\pi = SOS, h_1 = h', i = 1$

**repeat**

$i = i + 1$

$h_i = f_{trans}(h_{i-1}, S_{i-1}^\pi) \{update graph state\}$

$\theta_i = f_{out}(h_i)$

$S_i^\pi \sim \mathcal{P}_{\theta_i} \{sample node i's edge connections\}$

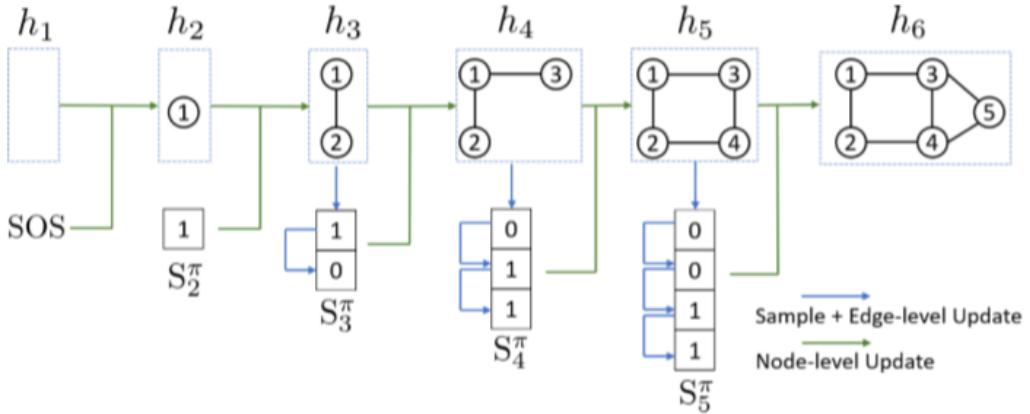
**until**  $S_i^\pi$  is  $EOS$

**Return**  $S^\pi = (S_1^\pi, \dots, S_i^\pi)$

---

那么剩下的就是如何去获得上面算法中即生成模型的参数了。GraphRNN 使用深度学习的方法来建模图生成的问题，不需要人为的去构建特征来生成指定特征的图，能以图数据为基础来生成类似的图，重点在于如何设计 GraphRNN 的模型结构。

如图 1 所示为 GraphRNN 生成图的过程。



*Figure 1.* GraphRNN at inference time. Green arrows denote the graph-level RNN that encodes the “graph state” vector  $h_i$  in its hidden state, updated by the predicted adjacency vector  $S_i^\pi$  for node  $\pi(v_i)$ . Blue arrows represent the edge-level RNN, whose hidden state is initialized by the graph-level RNN, that is used to predict the adjacency vector  $S_i^\pi$  for node  $\pi(v_i)$ .

Figure 1: GraphRNN 生成图的过程

GraphRNN 模型是基于 RNN 的模型，以 RNN 为基础，从一个起始状态，利用 RNN 逐步生成图的状态，每一步向图中添加一个结点，再利用 RNN 生成新结点与已存在结点的连接关系，接着再向图中添加结点，重复下去直至  $EOS$ 。结合之前的定义， $h_i$  就是在生成过程中图的状态，它是对已经生成的图的一个 encode 之后的结果，每步输出  $h_i$  后就会将  $h_i$  作为另一个 RNN 的输入—用于生成边的连接关系的 RNN。以上过程就是论文中提到的两个 RNN—graph-level RNN, edge-level RNN。结合生成图的算法，我们可以知道有这几个部分是比较关键的： $f_{trans}, f_{out}, \mathcal{P}_{\theta_i}$ 。论文中对这部分都是采用神经网络来实现的，在我理解中  $f_{trans}$  就是 graph-level 的 RNN，被  $\theta_i$  参数化的  $\mathcal{P}_{\theta_i}$ （论文中没有对其进行详细的说明， $\mathcal{P}_{\theta_i}$  具体是什么形式的呢？是全局共用一个  $\mathcal{P}_{\theta_i}$  吗？）

) 用于生成边的连接关系，即一个二进制序列。有一个值得注意的点： $S_i^\pi$  是一个不定长的序列，但是 RNN 的输入是定长的。论文中是这样处理的这个问题的：采用 BFS 遍历图，可以得到每个结点的依赖关系，通过对图数据集进行分析，可以得到每个节点依赖的已生成结点的数量  $M$  的分布，最终在训练时固定下  $M$ ，即在训练时每个结点所依赖的结点数是确定的， $M$  是个超参数。

那么如何进行训练呢？（这部分论文中并没有进行详细的描述，代码中有较详细的过程）首先要将传统格式的图数据转换为论文中定义的序列形式的图。对于每个图，会对其结点进行多次排列，得到多次结点的不同序列，然后根据 BFS 算法对其进行遍历，可以得到在不同排列下的图生成序列。这样一个图的概率  $p(G)$  就可以通过求联合分布的边缘分布得到了。其次，在训练的过程中，使用了 Teacher Forcing 的技术进行训练。

看看这篇论文是否解决了一开始所提到的几个难点呢？

- 生成空间太大且可变。个人理解论文中的 graph-level 和 edge-level 的 RNN 相当于对图进行了编码，能否看作是对整个图的嵌入呢？
- 使用序列来表示图的生成过程，将一个确定的图的概率表示为联合概率  $p(G, S^\pi)$  的边缘分布。
- 复杂的依赖关系。使用 BFS 遍历图，得到数据集中每个结点所依赖的结点的数量的分布，固定下来作为超参数。（**这是否可以作为一个改进的点呢？**）

未来工作的方向。生成更大规模的图，高效地生成指定条件的图。

其他参考资料：

1. [What's teacher forcing for Recurrent Neural Networks?](#)

## 2 How powerful are graph gnns?

论文地址: <https://arxiv.org/abs/1810.00826>

源码地址: [powerful-gnns](#)

关键词: GNN, WL graph isomorphism test

写于: 2020-09-24

论文 [62] 对以往的 GNN 模型的表现能力（区分能力）进行了理论上的分析，主要针对不同 aggregation 的特点进行了分析，并针对以往的 GNNs 的弱点，设计了 GIN(graph isomorphism network)，达到了与 WL sub tree 相匹配的性能。论文中使用两种分类任务对以往的 GNNs 和 GIN, WL sub tree 进行了测试：结点分类、图分类。

论文由一个问题开始讨论 GNNs 的区分能力。

给定两个图  $G_1, G_2$ ，不同的 GNNs 可能会将它们（或者两图中的某两个结点）嵌入到相同的表示。这是为什么呢？这就要讨论 GNNs 是如何来获得结点/图的 embeddings 了。考虑一个通用的 GNN 模型：

$$h_v^{(k)} = \text{COMBINE}(h_v^{(k-1)}, \text{AGGREGATE}(h_u^{(k-1)} | u \in \mathcal{N}_v))$$

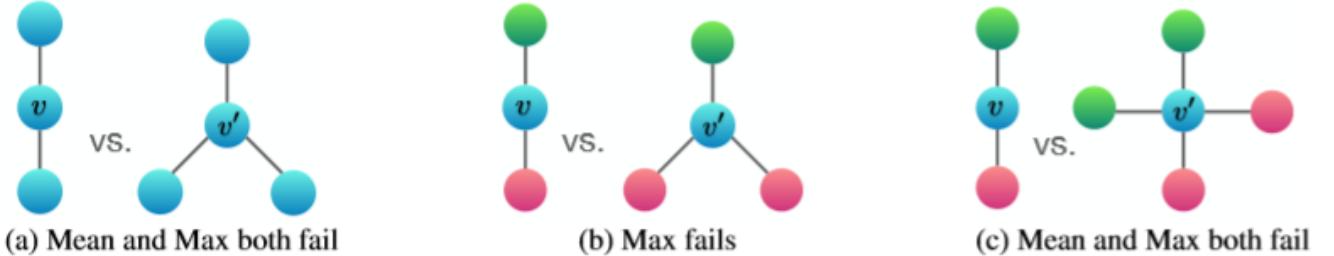
上式中的 AGGREGATE 用来聚集邻居结点的信息（如 mean, max, GRU, LSTM, GAT 等），COMBINE 则用来将聚集后的邻居结点的信息与结点自身的信息进行组合（如拼接。其实这很像一个图中的消息传播模型，现有的很多 GNN 模型也是基于消息传播的方法来汇聚邻居结点的信息， $k$  层的 GNN 相当于汇聚了来自  $k$ -hop 的邻居的信息。（**能否使用其他的框架来构建 GNN 模型呢？**）对于图  $G$  的 embedding，是由  $G$  中的所有结点形成的（这涉及 graph embedding 的问题，参见其他论文）。

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

再回到论文中的问题：**以往的 GNN 可能无法区分某些结点/图，即对不同的结点/图不能生成唯一的 embedding，不是 injective 的！**。根据结点/图的 embedding 生成的方法可以看出，关键在于

AGGREGATE, COMBINE, READOUT，论文中将这些定义为 *multi-set function*，multi-set 是一个可以有重复元素的集合，multi-set function 就是定义在 multi-set 上的函数。问题就出在了 GNNs 使用的某些 multi-set function 上！

论文主要对不同的 AGGREGATE 进行了分析，且主要分析了 mean, max 两种方法。如上图所示，展示了



**Figure 3: Examples of graph structures that mean and max aggregators fail to distinguish.** Between the two graphs, nodes  $v$  and  $v'$  get the same embedding even though their corresponding graph structures differ. Figure 2 gives reasoning about how different aggregators “compress” different multisets and thus fail to distinguish them.

Figure 2: 令 Mean, Max 失败的例子

mean, max 无法区分两个图中不同结点的邻居情况（neighborhood，即 AGGREGATE 后的结果是一样的）。论文中分别对 mean, max 的特性进行了阐述。

先给一个形式化的定义，假设  $X$  为某个结点的邻居结点集， $h(X)$  就是 AGGREGATE。

**MEAN Learns Distributions** 当  $h(X) = \frac{1}{|X|} \sum_{x \in X} f(x)$  时, 若对于不同的邻居结点集  $X_1, X_2$ , 若  $h(X_1) = h(X_2)$ , 则  $X_1$  和  $X_2$  有着相同的分布。从统计意义上理解, 相当于两个分布的均值 (mean) 相同。所以, 当我们更需要捕捉图中某种信息的分布或者信息重复较少时, mean 会有不错的效果。

**MAX Identity "Skeleton"** (妙!) 当  $h(X) = \max_{x \in X} f(x)$  时, 若对于不同的  $X_1, X_2$  有  $h(X_1) = h(X_2)$ , 则  $X_1$  和  $X_2$  有着相同的 underlying set。当使用 max 作为 AGGREGATE 时, 相当于对 multi-set 中的“重点”元素进行关注, 当放在整个图中看时, 相当于抽取了图中某种意义上比较“强”的结点。

接下来就是重头戏 — GIN 了。

论文中对结点和图的 GIN 定义如下:

$$h_v^{(k)} = MLP^{(k)}((1 + \epsilon^{(k)}) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)})$$

$$h_G = \text{CONCAT}(\text{READOUT}(h_v^{(k)} | v \in G) | k = 0, 1, \dots, K)$$

看到上面的定义, 你可能会疑惑, 为什么 GIN 就比其他的 GNNs 好呢? 这和以往的 GNNs 有什么区别呢? 论文中对 GIN 使用的 AGGREGATE 进行了理论上的证明, 证明 GIN 是 injective 的。

论文中通过两个结论证明了 GIN 的 injective 性。说了几次 injective 了, 那什么是 injective 呢?

**injective function:** 定义域中的每个不同的原像在值域中的像都是唯一的。那么, 为了让 GNN 有足够 powerful 的表现/识别能力, GNN 需要尽可能精确地区分每个不同的结点/图 (GNN 作为一种主要的 embedding 方法, 高精度的表示能力能为下游任务打好基础), 所以应尽量使 AGGREGATE, COMMBINE, READOUT 是 injective 的。

### 证明 GIN 的 injective

再来谈一下 Weisfeiler-Lehman (WL) graph isomorphism test。WL isomorphism test 是用来比较两个图是否是同构的。先解释一下同构的概念。

抛开图同构, 把同构概念单独拎出来看。对于两个系统中的对象, 同构映射能够保持两个系统中的对象一一对应, 且对象之间的关系也能够一一对应。图同构中,  $v_1 = f(u_1), v_2 = f(u_2), u_1, u_2 \in G_1, v_1, v_2 \in G_2$ , 若  $u_1, u_2$  之间有边, 则  $(v_1, v_2) = g((u_1, u_2))$ 。

WL test 是通过迭代的聚集邻居的标签, 每次迭代后将自己的标签和邻居们的标签映射成一个新的标签 (相当于 AGGREGATE 后在 COMBINE), 迭代完成后比较两个图的标签分布 (简单来说即各个标签分别有几个) 是否一致 (**结点的标签能否收敛呢? 为什么标签的分布能用来判定是否同构呢?** ), 如果标签分布一致, 则两个图可能同构的。WL subtree kernel[51] 是基于 WL test 的。我并没有通读论文, 但是粗略看了一下图, 感觉 WL subtree kernel 和现在的 GNN 已经很像了, 对于每个结点, WL subtree kernel 构建以该结点为 root 的树, 子节点为其邻居结点, 子节点的子节点是邻居结点, 如此循环定义, 最后以这棵树作为结点标签的依据。WL subtree kernel 是目前基于聚合方式的 GNNs 的性能上界 (证明见论文 appendix)。但是 WL subtree kernel 并不能结合结点的信息, 对于现在图中结点丰富的信息来说, 这不免是有点浪费的, 而且, 个人认为 WL subtree kernel 捕捉的是图的结构 (毕竟是衍生于图同构算法), 或许图中的其他信息它是没有捕捉到的 (**是什么信息呢?**)

关于未来的研究方向。有没有比基于聚集邻居结点信息 (信息传递) 方法更好的一个 GNN 框架呢?

关于 WL test 可参考的博客/非文献资料:

1. [The Weisfeiler-Lehman Isomorphism Test](#)

### 3 Gated Graph Sequence Neural Networks

论文地址: <https://arxiv.org/abs/1511.05493>

源码地址: [ggnn](#) (coded in Lua)

slides: [ggnn-talk](#)

关键词: RNN, Graph Sequential tasks

写于: 2020-09-28

这篇论文 [28] 提出了一种基于 GRU[9] 的 GNN，能够进行输出单个值的任务（如结点分类、图分类等），也能完成序列输出的任务（如最短路、欧拉环等）。论文中使用 bAbI[59]（bAbI 是 Facebook AI 推出的文本理解/推理任务生成器）任务和程序验证的任务对 GG-NN, GGS-NN 进行了测试均达到了很好的效果。

论文中讨论的是针对有向图的 GNN。那么是如何表示有向图的 GNN 呢？

对于有向图  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ，其邻接矩阵由两部分组成  $A = [A_{in}, A_{out}]$ ,  $A_{in}$   $A_{out}$  分别为入，出邻接矩阵。 $\mathcal{G}$  中每个结点  $v$  都定义了其 IN 和 OUT 结点集，分别表示指向  $v$  的边的起始结点集和从  $v$  出发的边的终点结点集。 $v$  的邻居结点集 NBR 定义为  $IN \cup OUT$ ，并且对于边和结点都可以有各自的标签。

论文中对 GNN 进行了回顾。GNN 将图数据映射到输出的过程中，可以划分为两个部分：PROPAGATION MODEL, OUTPUT MODEL，分别用来计算节点的表征和将节点表征映射到输出。

论文针对非序列的输出和序列的输出分别提出了 GG-NNs(Gated Graph Neural Networks) 和 GGS-NNs(Gated Graph Sequence Neural Networks)。其中 GG-NNs 是基于 GNN 的，不同的是使用了 GRU 来构建 PROPAGATION MODEL 和 OUTPUT MODEL。用于输出序列的 GGS-NN 则是在 GG-NN 的基础上构建的。接下来先介绍一下 GRU[9]，在介绍 GG-NN 和 GGS-NN。

**GRU(Gated Recurrent Unit)** 门控循环单元。GRU 可以视为 LSTM 的变体，与 LSTM 很相似。与常规的 RNN 中的单元相比，门控神经单元间的连接是不变的，但每个门控神经单元内部都是经过精心设计的。作为门控 RNN 能够学会决定何时清除状态。

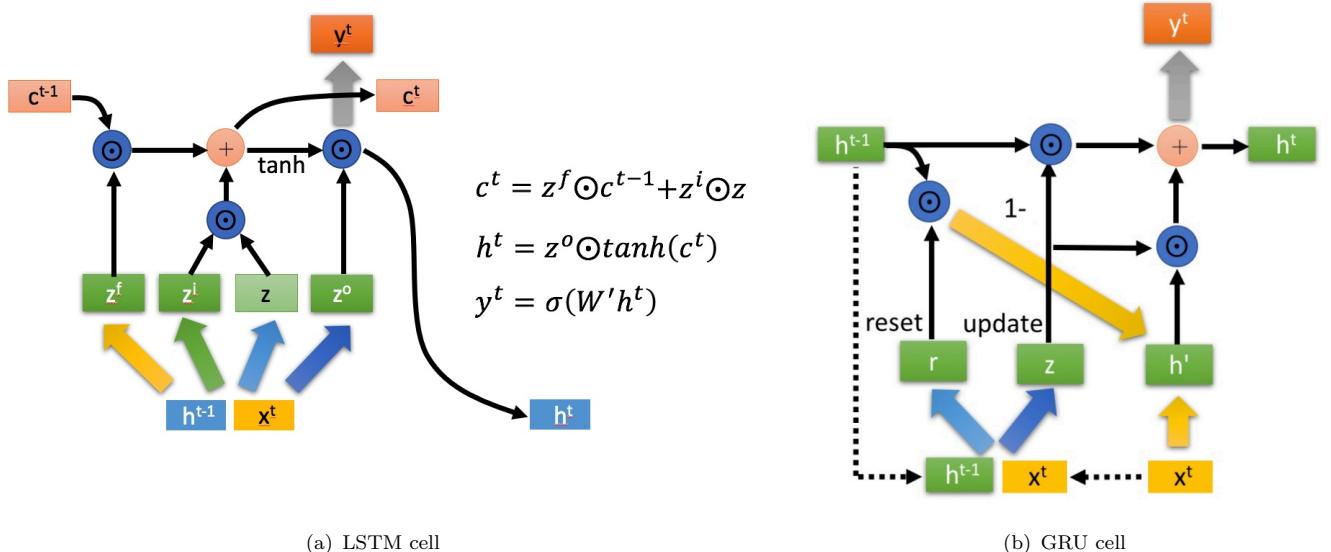


Figure 3: LSTM 和 GRU cell 的内部结构

**GG-NN** GG-NN 的过程如下图所示。从 Fig.4 中可以看出，GG-NN 的输入是”Annotate nodes with problem

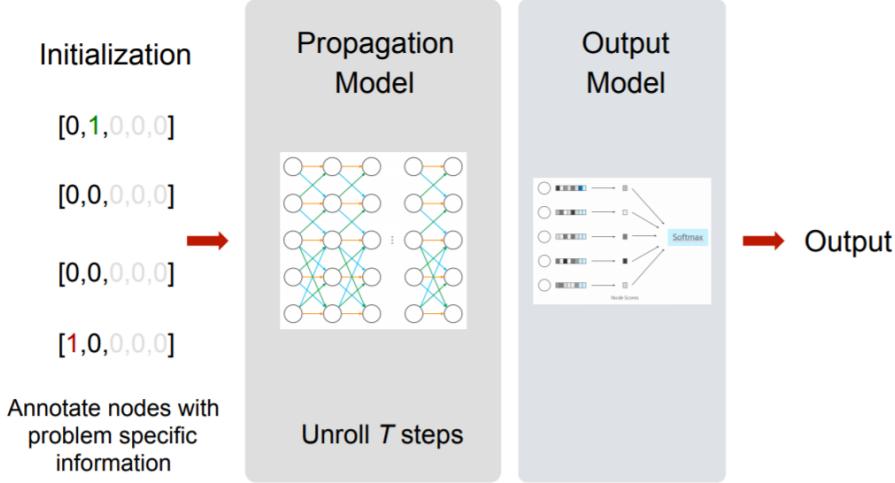


Figure 4: GG-NN

specific information”，在论文中称为”Node Annotations”，这与之前提到的结点标签并不是同一个，node annotations 是在特定问题下所定义的“标签/特征”。根据特定的问题领域会为每个节点生成 annotation（[如何生成 annotations?](#)），在输入层会将 annotation 用 0 填充成固定的大小作为网络的输入。结点的 annotations 通过 PROPAGATION MODEL—基于 GRU 的 t 层（[论文中成 t-steps, 但根据我的理解是指网络有 t 层](#)）网络，再将计算后的数据通过 OUTPUT MODEL 输出任务结果。

**GGS-NN** GGS-NN 的过程如下。从 Fig.5 中也能看出，GGS-NN 是在 GG-NN 的基础上搭建的。GGS-NN 的

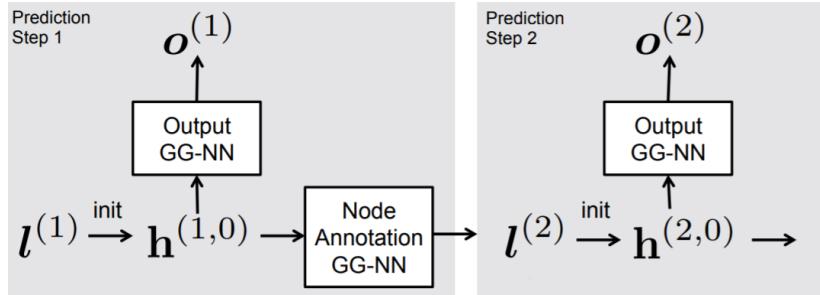


Figure 5: GGS-NN

输入也是 Node Annotations。每一步输入的 Annotations 是根据上一步的的隐层的输出转换而来的。每一步中的隐层输出会分别传给两个 GG-NN，一个用来产生这一步的输出，另一个则用来产生下一步的输入，即下一步输入的 node annotations。其实，在 GGS-NN 中除了上述两个 GG-NN 外，还有一个 GG-NN 用于在每一步确定是否继续，该 GG-NN 会在 graph-level (把图看成一个特殊的结点，与图中的所有结点都有联系) 上来进行一个二分类。

论文中使用 bAbI 任务和程序逻辑验证进行测试。bAbI 任务中将实体和实体间关系看作点和边（有点类似知识图谱），利用 GG-NN/GGS-NN 来进行推理。为解决程序验证中的 program invariants 问题是这篇论文的一个主要出发点。通过将程序运行过程中 heap 的状态看作图数据，在这些数据的基础上以序列的方式生成程序的 separation logic[42] 表达式。

网络参考资料：

1. [门控循环单元 \(GRU\) 的基本概念与原理](#)
2. [Illustrated Guide to LSTM's and GRU's: A step by step explanation](#)

## 4 Graph Structure of Neural Networks

论文地址: <https://arxiv.org/abs/2007.06559>

源码: [graph2nn](#)

slides: [Graph Sturcture of Neural Networks](#)

关键词: **Neural Networks, Graph Analysis**

写于: 2020-10-14

这篇论文 [66] 从图的角度来解释神经网络 (Neural Networks, NN)，用 relational graph 来对 NN 进行建模，并在 relational graph 的基础上定义了 NN 训练的过程。在最短路径和聚集系数的基础上分析 NN 的性能。以这样的视角来解释和分析 NN 是一个新颖的角度，**为设计 NN 和解释 NN 提供了新的思路**。

**如何建模 NN 为图？** 引用文中的一句话: focus on message exchange, rather than just on directed data flow。建模的关键出发点是对 NN 中神经单元之间信息的交换，而不是信息的流向。想象如果给你一个图  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ ，怎么把它看作一个 NN 呢，NN 的训练如何在其上展开呢？论文中把  $\mathcal{G}$  称为 relational graph。一个 NN 会有多层，可是一个图只有这么多结点，那么怎么训练这个“图化”后的 NN 呢？论文中使用 round 来对应 NN 中的层。则 NN 中信息在层间传递时，在  $\mathcal{G}$  中看起来是这样的：

$$\mathbf{x}_v^{(r+1)} = AGG^{(r)}(f_v^{(r)}(\mathbf{x}_u^{(r)}, \forall u \in N(v)))$$

其中， $\mathbf{x}_v^{(r)}$  是结点  $v$  在第  $r$  round 时的特征， $N(v)$  是结点  $v$  的邻居 (relational graph 中的结点都是自环的)。AGG 类似于 NN 中的激活函数， $f$  则类似于 NN 中的对上一层的输入进行线性组合或这卷积操作等。这看起来和基于消息传递的 GNN 模型看起来是很相似的。

论文中基于 relational graph 对现有的一些 NN 进行了分析，如固定宽度的 MLP、可变宽度的 MLP、ResNet 的一些变体等，使用 relational graph 进行训练，并分析了 NN “图化”后的结构与性能之间的关系。

### 方法解决的问题/优势

- 开创性的将 NN 视作图数据进行分析 (哎，我也想到了这个点子)
- 以图的视角分析 NN 或许能够对 NN 的设计和理解产生很大影响

### 方法的局限性/未来方向

- 个人认为论文中对 NN 建模为 relational graph 的想法很不错，也很符合现在的 GNN 做法，但是对于复杂的 NN，以及 NN 中的一些特殊的操作，论文中的 relational graph 就有点简单了，**难以表现复杂的 NN 结构**
- 对 relational graph 的分析还比较少，局限于最短距离和聚集系数，可以分析图中的局部性的结构，如团、子图、motif 等，以及一些其他的角度
- **有期望能够打通 GNN 和 NN 之间的联系，或许在此之上能够提出新的深度学习模型!!!**
- 在论文基础上对 NN 进行解释

## 5 Variational Graph Auto-Encoders

论文地址: <https://arxiv.org/abs/1611.07308>

源码: [gae](#)

关键词: **Variational auto encoders**

写于: 2020-10-16

该论文 [26] 利用 VAE (Variational auto-encoder[25]) 来学习无向图的隐表示，并可以从隐表示解码出原来的图。

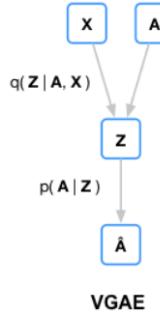


Figure 6: VGAE

**VGAE 思路** 模型的思路如图 Fig.6所示。其中  $X \in \mathbb{R}^{N \times D}, A \in \mathbb{R}^{N \times N}$  分别是无向图  $G$  的结点特征矩阵、邻接矩阵， $Z \in \mathbb{R}^{N \times F}$  是  $G$  的隐表示， $\hat{A} \in \mathbb{R}^{N \times N}$  是通过隐表示解码出的图的邻接矩阵，相当于根据隐表示重建后的图的邻接矩阵。整个模型可以看作一个 VAE 模型，分为编码和解码两个阶段。从  $(X, A)$  到  $Z$  通过 GCN 来完成，从  $Z$  到  $\hat{A}$  则是通过隐表示间的内积再通过激活函数得到边的概率。

### 方法解决的问题/优势

- 提出使用 VAE 模型对图进行编码和解码
- 在对图进行编/解码的过程中，隐表示  $Z$  可以视作结点表征

### 方法的局限性/未来方向

- 个人感觉编码过程略粗糙，不精细
- 扩展性不好

在看这篇论文之前，我以为这篇论文的重点是图的重建，但是文中更多地体现的是获得隐表示，将隐表示用于下游任务。那么，**能否学习一个图编码器，能够很好地对图进行压缩，重建时又可能准确呢？**

## 6 Hierarchical Graph Representation Learning with Differentiable Pooling

论文地址: <https://arxiv.org/abs/1806.08804>

源码: [diffpool](#)

关键词: GNN, Graph Classification,  
Graph Pooling

写于: 2020-10-21

该论文 [65] 解决的是图分类的问题，针对之前的图分类中没有利用层次化的图的信息，提出了 DIFFPOOL (Differentiable Pooling) ——一种图池化 (graph pooling) 的方法，能够生成层次化的图的表示。DIFFPOOL 的思路在 GNN 的基础上，学习软聚类分配 (soft cluster assignment) 的矩阵，将每个结点分配到不同簇中形成一个新的图，在新的图上继续使用 GNN 和 DIFFPOOL 来得到图层次化的表示，最终图将池化为一个点，使用这个点的表征进行分类。

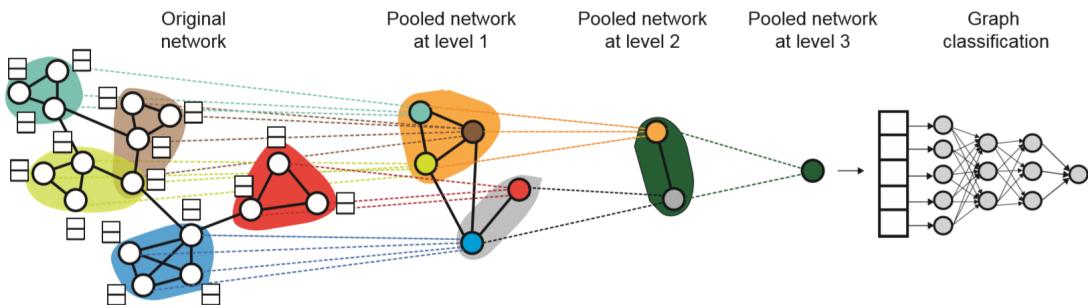


Figure 7: Overview of DIFFPOOL

**DIFFPOOL 思路** DIFFPOOL 的过程如 Fig.7 所示。通过逐步将图中的结点压缩成一个节点，形成新的图的方法，逐步得到图的多个层次化的表示，最终图将被压缩成一个结点，再将这个结点的表征输入 MLP 中对图进行分类。为了得到图的结点表征和对图中结点进行软聚类，DIFFPOOL 中一共用到了两个 GNN， $GNN_{embed}$  用于学习图的结点表征， $GNN_{pool}$  用于学习软分配矩阵。其中的软分配矩阵的作用就是用于把  $GNN_{embed}$  的结点表征矩阵压缩成结点更少的图，并把邻接矩阵压缩成对应的图的连接方式。现有图  $G = (X, A)$ ，DIFFPOOL 的过程形式化表示为：

$$\begin{aligned} Z^{(l)} &= GNN_{embed}^{(l)}(A^{(l)}, X^{(l)}) \\ S^{(l)} &= \text{softmax}(GNN_{pool}^{(l)}(A^{(l)}, X^{(l)})) \\ X^{(l+1)} &= S^{(l)} Z^{(l)} \\ A^{(l+1)} &= S^{(l)\top} A^{(l)} S^{(l)} \end{aligned}$$

上式中， $l$  表示 GNN 的第  $l$  层。在实际训练 DIFFPOOL 的过程中，论文中不仅使用了图分类作为学习的目标，同时使用了连接预测来监督模型的学习，为的是尽量使有边连接的结点被池化到同一个超点中。

### 方法解决的问题/优势

- 提出了一种新的图池化方法，使得处于类似簇/社群中的结点被压缩成一个超点
- 能够学习到层次化的图表征
- 能间接得到图的社群结构

## 方法的局限性/未来方向

- 使用其他的池化方法，减小计算代价
- 不太理解其中  $A^{(l+1)} = S^{(l)^\top} A^{(l)} S^{(l)}$  的原因
- 因为结点表征和邻接矩阵使分开来池化的，这样能保证结点表征和邻接矩阵的匹配吗

## 7 Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation

论文地址: <https://arxiv.org/abs/1806.02473>

源码: [rl\\_graph\\_generation](#)

关键词: Graph Generation, Molecular Graph, Reinforcement Learning

写于: 2020-10-21

该论文 [67] 主要针对的是指定目标下的图生成问题，论文的主要面向领域是药物发现，目的是生成满足某种属性以及约束条件的图。针对这个问题，论文提出了 GCPN (Graph Convolutional Policy Network) —— 一种强化学习模型，通过策略梯度 (policy gradient) 来优化指定领域的奖励和对抗损失，在领域特定的约束环境中生成符合要求的图。

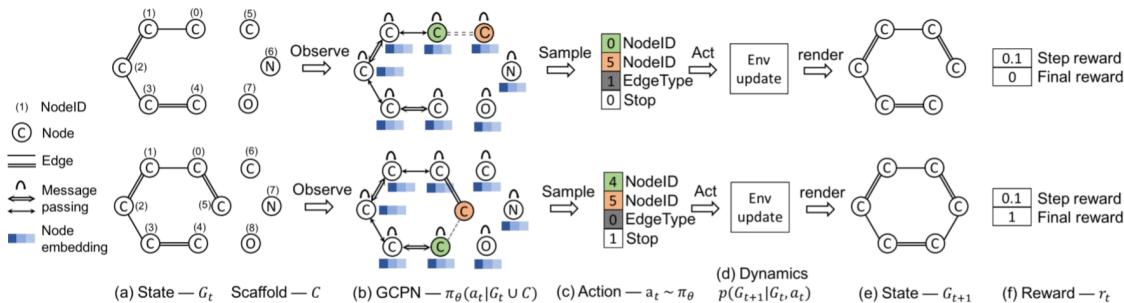


Figure 8: Overview of GCPN

**GCPN 思路** 整体过程如 Fig.8 所示。在生成一个分子的过程中，是逐步通过将子结构连接到分子中或者添加边来完成的。生成一个有效分子的过程可以视作马尔可夫决策过程，从最起始的分子开始，通过执行动作进行状态转移，每个分子就是一个状态。其中，最重要的就是如何选取动作，即在每个状态下执行什么样的动作:  $p(a_t | s_0, \dots, s_t)$ ，那么下一个状态就是:  $p(s_{t+1} | s_0, \dots, s_t, a_t)$ 。论文中通过策略网络 (policy network) 来模拟  $p(a_t | s_0, \dots, s_t)$ 。生成一个动作  $a_t$  后，会经过一些药物/化学方面的检验，来决定是否执行  $a_t$ ，如果执行则会增加 reward (即强化学习中的 reward)。

需要强调的是，GCPN 是通过 RL 来训练的，训练时不仅要考虑生成的动作是否符合要求还要考虑最后生成的分子是否满足某些性质及满足的程度。

## 方法解决的问题/优势

- 以目标导向的图生成问题来解决药物合成的问题，并且能满足药物需要满足的性质

- 以 RL agent 来对药物空间进行探索，生成的药物不一定与训练数据的分布一致，因为新的分子可能与训练数据的分布有较大差别，但是满足分子应有性质
- 将不可微的分子的属性和一些先验条件加入到模型中 监督模型的训练，使模型向期望的方向发展

### 方法的局限性/未来方向

- 可以将模型应用到更广的范围，按照先验条件生成符合指定条件的图数据

## 8 Auto-Encoding Variational Bayes

论文地址: <https://arxiv.org/abs/1312.6114>

来源: CoRR, 2013

作者: Diederik P Kingma, Max Welling

关键词: **Variational Bayes, Auto-Encoding, Bayes Inference, Probabilistic Model**

写于: 220-11-09

论文 [25] 为 bayes 概率图模型难以求解的问题提供了一种有效的思路，利用 auto-encoding 方法结合 variational lower bound 求解 bayes 图模型隐变量的后验分布。

在推断和学习中，我们可以认为数据是根据某个隐变量生成的 — 可以从数据得到隐变量，也能够根据隐变量得到数据 — 就像一个编码、解码的过程。但是，通常隐变量的分布是未知的、复杂的，难以通过假定一个已知的分布来近似隐变量的真实分布。

简单谈一下个人对隐变量的理解：宽泛的讲，隐变量就像现实背后的某种神秘的、未知的因素，我们所见的现实背后蕴含着这种神秘的因素 — 隐变量，但我们通常是看不到隐变量的，也无法对它进行直接的观测，比如直接描述它是什么、它的形式、它发生的概率等等。但当我们得知隐变量后我们就像获得了某种神奇的力量，知道隐变量发生后现实中会发生什么。从数据的角度来看，隐变量可以是数据背后的“真理” — 特征、数据产生的原因。当我们能够描述隐变量后，我们就能够根据隐变量生成我们想要的数据。这就像一个编码-解码的过程，将原始数据编码为特征，再根据特征产生数据。

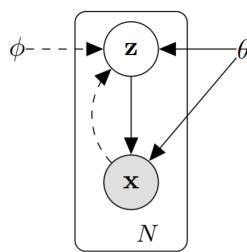


Figure 9: 有向图模型

### 那么问题来了：**如何得到隐变量呢？**

如图9所示，图中实线可以看作生成（即解码）过程 —  $p_\theta(z)p_\theta(x|z)$ ，虚线可以看作编码过程 —  $q_\phi(z|x)$ 。 $\theta$  表示真实的分布的参数， $\phi$  表示近似 ho 后验分布的参数 — 该近似分布用于近似真实的后验分布  $p_\theta(z|x)$ 。为了使近似分布  $q_\phi(z|x)$  和真实分布  $p_\theta(z|x)$  尽量相同，可以使用 KL 散度进行衡量。

$$\begin{aligned}
KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}] \\
&= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{z}|\mathbf{x})] \\
&= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log \frac{p_\theta(\mathbf{z}, \mathbf{x})}{p_\theta(\mathbf{x})}] \\
&= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{z}, \mathbf{x}) + \log p_\theta(\mathbf{x})] \\
&= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{z}, \mathbf{x})] + \log p_\theta(\mathbf{x})
\end{aligned} \tag{1}$$

由公式 (1) 可得，对于训练数据中的每一个样本有：

$$\log p_\theta(\mathbf{x}^{(i)}) = KL(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$$

其中，

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = E_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[-\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) + \log p_\theta(\mathbf{z}, \mathbf{x}^{(i)})]$$

因为 KL 是大于等于零的，那么显然有这样的关系： $\log p_\theta(\mathbf{x}^{(i)}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$ 。所以  $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$  又叫做变分下界 (variational lower bound)。 $\log p_\theta(\mathbf{x}^{(i)})$  相当于样本的对数似然函数，当给定了数据后，其实  $\log p_\theta(\mathbf{x}^{(i)})$  应该是确定的，那么为了使近似分布尽量接近真实分布，那么则应该让变分下界尽可能的大，这样近似分布就会尽可能地接近真实分布。OKAY！现在问题转化成了最大化  $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$  了。

那么问题又来了：**如何最大化变分下界呢？**

经过一顿操作后，变分下界可写为：

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = -KL(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + E_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})]$$

其中第一项可以看作正则化项（当增加隐变量的数量时，可以防止过拟合），第二项可以看作重构损失。为了最大化变分下界，可以使用梯度下降的方法，但是上式中变分下界难以计算，且近似分布  $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$  是未知的。如果直接使用蒙特卡洛方法计算变分下界，将会带来较大方差。论文中针对这个问题使用了重参数化 (reparameterization) 来表示隐变量： $\hat{\mathbf{z}} = g_\phi(\epsilon, \mathbf{x})$ ，其中  $\epsilon$  服从于某个分布  $p(\epsilon)$ 。那么问题就转化成了选择合适的函数  $g$  和分布  $p(\epsilon)$ 。已知一个函数关于其自变量的后验分布的蒙特卡洛估计为下式：

$$E_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[f(\mathbf{z})] = E_{p(\epsilon)}[f(g_\phi(\epsilon, \mathbf{x}^{(i)}))] \simeq \frac{1}{L} \sum_{l=1}^L f(g_\phi(\epsilon^{(l)}, \mathbf{x}^{(i)}))$$

其中  $L$  为对每个数据  $\mathbf{x}^{(i)}$  进行蒙特卡洛采样的次数。那么使用蒙特卡洛估计重参数化后的变分下界，形式为：

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^L \log p_\theta(\mathbf{z}^{(i,l)}, \mathbf{x}^{(i)}) - \log q_\phi(\mathbf{z}^{(i,l)}|\mathbf{x}^{(i)})$$

其中， $\mathbf{z}^{(i,l)} = g_\phi(\epsilon^{(i,l)}, \mathbf{x}^{(i)})$ ,  $\epsilon^{(l)} \sim p(\epsilon)$ 。之后便可以使用小批量的随机梯度下降算法优化参数  $\theta, \phi$ 。

关于 VAE 的 tutorial 可以参考：[Tutorial on Variational Autoencoders](#)。未使用重参数化技巧与使用了重参数化技巧对比如 Fig.10[14] 所示：

一些网上的参考资料：

- [变分推断](#)

## 方法解决的问题/优势

- 从参数化变分下界，使得能够使用随机梯度下降算法优化参数

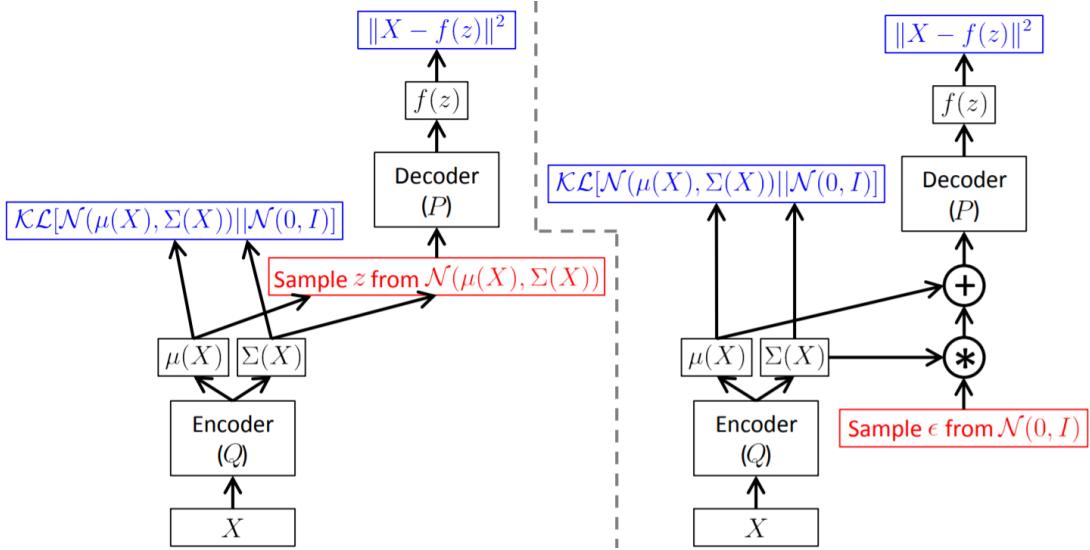


Figure 10: without vs. with reparameterization

### 方法的局限性/未来方向

- 时序模型
- 具有隐变量的监督学习模型，可用于学习复杂的噪声分布

## 9 Deep Graph random Process for Relational-Thinking-based Speech Recognition

论文地址：<https://smcnus.comp.nus.edu.sg/wp-content/uploads/DGP.pdf>

slides：[slides](#)

关键词：ASR, Relational Thinking, Deep Graph

写于：2020-12-10

该论文 [18] 针对语音领域中如何对感知 (percept) 建模，并将感知融入到语音相关的问题中进行了探索，提出了一个贝叶斯非参数化的深度学习方法 (Bayesian nonparametric deep learning method) — Deep Graph random process (DGP)，使用 Graph 对感知进行建模。

**问题定义** Relational Thinking 是人类学习过程中至关重要的一个过程，在学习的过程中，我们会接收到各种各样的信息，不管是来自视觉、听觉、触觉等，但是我们不会有意识地去保存它们（这是与 Relational Reasoning 的主要区别，relational reasoning 是有意识地处理这些 relation information）。在各种各样的信息之间隐藏着联系，这些信息及它们之间的联系组成了我们的 percepts (感知)。论文针对语音/对话中信息之间隐藏的 relation 进行研究。

在通常的 ASR (Automatic Speech Recognition) 任务中，整个任务被分为两个过程：语音建模、语言解码，即以模式匹配的方法来解决这个问题。这种方法有一个很明显的缺点：没有考虑 relational information，当然，如何将 relational thinking 融入到 ASR 中是很困难的。**在交谈过程中，我们形成的感知是无意识的、无穷的**。有一些研究已经表明，在问答系统中，即使只将前一段对话的内容考虑到语音模型中，也会大大地提高模型的效果。但**对 percepts 进行建模是很困难的** — 这也是论文重点解决的问题。

目前，混合的语音循环神经网络隐马尔可夫模型（RNN-HMM）在许多方面仍优于端到端的编码-解码的方法，而且也越来越流行。RNN 确实能够捕捉时间上的依赖关系，但对于复杂的关系（如 relational thinking）则表现得不足。Graph 能够描述更复杂的关系。但是依然存在一些挑战：

- Graph 相关的方法通常要求输入的数据是 Graph 形式的数据。但是在语音中并没有直接的图数据
- 目的是对 relational thinking 进行建模。但是在语音中 percepts 是隐含的

**DGP 思路** DGP 将 relational thinking 中的 percepts 建模为概率图，并且不需要任何关系数据。接下来更具体地描述 DGP 是如何对 percepts 进行建模的。

在语音中，处理的基本单位可以是音频数据的一个采样点、语音帧、语音片段（utterance）、一段语音等。论文中考虑的 utterance，一个 utterance 可以是一句话的发音。在 DGP 中，给定一个 utterance 以及它之前的 utterances（即历史 utterances），可以生成无数个 percept graphs，这与之前说到的 relational thinking 是相对应的。当前 utterance 和历史 utterances 之间不同的连接关可以看作一个 percept（也许这就是它们称作 **percept graphs** 的原因吧。以 utterances 连接成的 Graph 来定义 percept）。为什么是无数个 percept graph 呢？可以从直觉上来理解，在我们接收信息时，信息之间并没有生成稳定的信息，各个信息之间都是会产生关联的，并且基于我们的先验知识和历史信息，会在脑海中产生各种各样的理解 — 相当于接收到的信息的不同组合，而这个组合是有无限可能的。并且，由于这些 percepts 是无意识的，percept graph 中边的概率是接近于 0 的。

在 DGP 的 percept graph 中，Graph 中的结点表示 utterance，边表示结点之间的关系。论文中假设边的取值都服从 Bernuli 分布，且边的取值（即边存在的概率）是趋近于 0 的。看到这里貌似解决了对 percepts 建模的问题，但是该怎么解决涉及无数个 percept graphs 的问题呢？论文中对无数个 percept graphs 进行了 transform，在 percept graphs 的基础上生成 **summary graph**。获得 summary graph 后即可将 graph embedding 作为语音模型的额外输入来提高模型性能。那么如何操作呢？

**实现细节** 先定义一些形式化的表示。 $U_i$  表示 utterance  $i$ ， $U_{i-o:i-1}$  表示  $U_i$  之前的  $o$  个 utterances。 $U_i$  的特征是一系列的语音特征  $\mathbf{X}_i = \{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \dots, \mathbf{x}_{i,T}\}$ ，对应的标签为  $\mathbf{Y}_i = \{\mathbf{y}_{i,1}, \mathbf{y}_{i,2}, \dots, \mathbf{y}_{i,T}\}$ 。 $\{\alpha_{i,j}^{(k)}\}_{k=1}^{+\infty}$  表示第  $k$  个 percept graph 中  $U_i, U_j$  间边存在的概率。 $\{U_i\}_{i=1}^M$  是用于训练模型的  $M$  个 utterances。

**Deep Graph Random Process** Deep Graph Random Process 是一个随机过程，描述了一个机制，该机制主导无穷个概率图生成 — 即无穷个 percept graphs 的生成。在 percept graph 中，每个结点是 utterance 的 embedding：

$$\mathbf{v}_i = f_\theta(\mathbf{X}_i)$$

其中  $f_\theta$  是 NN (Neural Network)。DGP 的核心是一系列的深度 Bernuli 过程 (DBP)，每一个 DBP 负责一对结点间边的生成，即：

$$\{\alpha_{i,j}^{(k)}\}_{k=1}^{+\infty} \sim DBP(Bern(\lambda_{i,j}))$$

其中  $Bern(\lambda_{i,j})$  表示结点  $i, j$  间服从概率为  $\lambda_{i,j}$  的 Bernuli 分布，所有的 percept graphs 的边的概率都是从对应的 Bernuli 分布中采样的！

**Coupling of Innumerable Percept Graphs** 这一步的目的是如何生成 summary graph，因为 summary graph 本身也是 probabilistic graph，而且结点已知，那么只剩节点之间边的概率了。其实从上一步就可以看出，单个 percept graph 中边是服从 Bernuli 分布的，单考虑一条边的话，summary graph 中对应边就是重复的 Bernuli 抽样，所以可以把 summary graph 中的边看作是服从 Binomial 分布的。那么 summary graph 中边的概率为：

$$\tilde{\alpha}_{i,j} = \sum_{k=1}^{+\infty} \alpha_{i,j}^{(k)}, \quad \tilde{\alpha}_{i,j} \sim \mathcal{B}(n, \lambda_{i,j})$$

但上式存在一个问题， $n \rightarrow +\infty$ ,  $\lambda_{i,j} \rightarrow 0$ , 那么该如何解决这个问题呢？

**Inference and Sampling of Edges of Summary Graph** 为了采样得到 summary graph 中边的概率，论文中采用一个高斯分布来近似上文提到的 Binomial 分布。对于每条边，近似的高斯分布为：

$$\mathcal{N}(m_{i,j}, m_{i,j}(1 - m_{i,j})), \quad m_{i,j} = n\lambda_{i,j}$$

**Application of DGP for Acoustic Modelling** 为了从 summary graph 中提取出更具有代表性的信息，或者说为了更好地适应下游的任务，还需要对 summary graph 进行一个转换。转换的方式是赋予每条边一个权重，生成新的图 — task-specific graph (这样做的原理论文中并没有给出)。这很像注意力机制。权重是从一个高斯分布中采样得到的，那么 task-specific graph 的边表示为：

$$\bar{\alpha}_{i,j} = s_{i,j} * \tilde{\alpha}_{i,j}$$

上式中  $s_{i,j}$  是对应边的权重，论文中对权重服从的高斯分布进行了一定的假设：权重是以 summary graph 边的取值为条件的，即：

$$s_{i,j} | \tilde{\alpha}_{i,j} \sim \mathcal{N}(\tilde{\alpha}_{i,j} * \mu_{i,j}, \tilde{\alpha}_{i,j} * \sigma_{i,j}^2)$$

得到 task-specific graph 后，即可按照 GNN/GCN 的方式获得 graph embedding (注意：每处理一个 utterance 时都会有对应的 task-specific graph)。接下来就是应用上述一系列工作的结果 — graph embedding 了。论文中将整个框架称作 relational thinking network (RTN)，使用 simple recurrent unit (SRU) 作为构建单元。具体的可以参考 SRU，说明一下如何在 acoustic model 中使用 graph embedding：将 graph embedding 与原来的输入进行拼接即可。

**Learning** 知道了上述步骤，那么如何来学习整个模型呢？论文中使用 variational inference 来联合训练 DGP、task-specific graph 中边的权重、acoustic model。这个可以参考 [25]，目标函数是 evidence lower bound(ELBO)：

$$\sum_{i=1}^M \left\{ \text{KL} \left( q \left( \tilde{\mathbf{A}}, \mathbf{S} | \mathbf{X}_{i-o:i} \right) \| p \left( \tilde{\mathbf{A}}, \mathbf{S} | \mathbf{X}_{i-o:i} \right) \right) - \mathbb{E}_{\tilde{\mathbf{A}}, \mathbf{S}} \left[ \log P \left( \mathbf{Y}_i | \mathbf{X}_i, \tilde{\mathbf{A}}, \mathbf{S} \right) \right] \right\}$$

其中， $\tilde{\mathbf{A}} = [\tilde{\alpha}_{i,j}]$ ,  $\mathbf{S} = [s_{i,j}]$ 。上式可以转化为：

$$\begin{aligned} & \sum_{(i,j) \in \tilde{E}} \left\{ \text{KL} \left( \mathcal{B} \left( n, \tilde{\lambda}_{i,j} \right) \| \mathcal{B} \left( n, \tilde{\lambda}_{i,j}^{(0)} \right) \right) \right. \\ & \quad \left. + \mathbb{E}_{\tilde{\alpha}_{i,j}} \left[ \text{KL} \left( \mathcal{N} \left( \tilde{\alpha}_{i,j} \odot \mu_{i,j}, \tilde{\alpha}_{i,j} \odot \sigma_{i,j}^2 \right) \| \mathcal{N} \left( \tilde{\alpha}_{i,j} \odot \mu_{i,j}^{(0)}, \tilde{\alpha}_{i,j} \odot \sigma_{i,j}^{(0)^2} \right) \right) \right] \right\} \end{aligned}$$

但并不能直接最大化上式来作为优化目标，因为其中会涉及到  $n \rightarrow +\infty$ ，所以还需要对其进行转化，具体的转化过程可以参考论文中的证明和附带材料。

Okay，整个过程就是这样了，整体框架如 Fig.11 所示。论文中还对 RTN 的具体实现进行了介绍，包括上文中涉及到的一些参数的计算，如  $m_{i,j}, \mu_{i,j}, \sigma_{i,j}$  等，以及模型的网络结构等。

## 方法解决的问题/优势

- 提出了一种对 percept 建模的方法，percept graphs 的想法很棒
- 以 graph 的形式表示 percepts，并无需 relation data 就可以学到隐含的 relational thinking
- 将 relational thinking 融入到 acoustic model 中

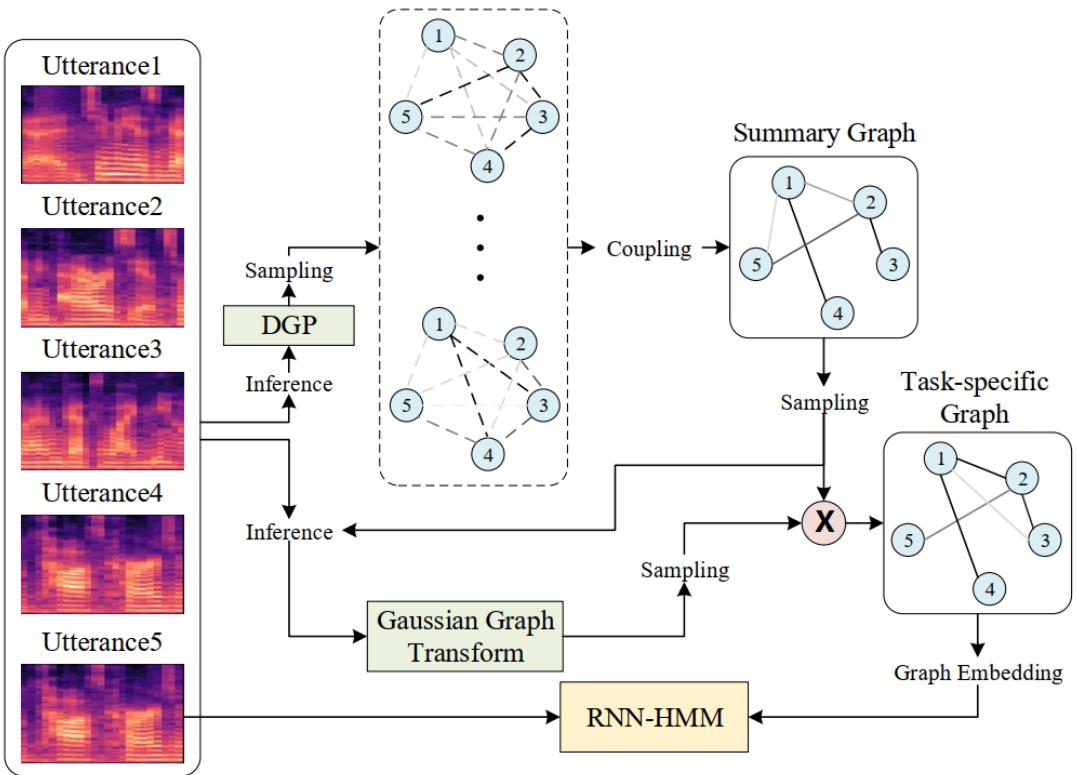


Figure 11: Architecture of RTN for acoustic modelling

### 方法的局限性/未来方向

- 个人认为 variational inference 会涉及到较多的运算
- 可以将 percept graphs 的想法应用到更多的问题上**
- task-specific graph 的由来没有解释清楚
- 论文中只结合了历史的 utterances，并没有结合先验知识**。尽管不同的人先验知识是不同的，但是在某个特定的应用场景下，先验知识是有范围的，可以借助知识图谱来表示先验知识，再结合本文的方法完成特定场景下的对话任务。**这也可看作是一个多模态信息融合的方法**
- 在生成 summary graph 时，相当于给所有的 percept graphs 分配了相等的权重（注意力），然而在实际的思考过程中，**不同的 percepts 应该有不同权重的，在生成 summary graph 时对不同的 percept graphs 赋予不同的注意力是否更合理？**

## 10 Adversarial Autoencoders

论文地址: <https://arxiv.org/abs/1511.05644>

来源: ICLR, 2016

作者: Alireza Makhzani ,Jonathon Shlens ,Navdeep Jaitly ,Ian Goodfellow ,Brendan Frey

源码: [AAE\\_pytorch](#)

关键词: GAN, Autoencoder, Generative Models

写于: 2020-12-20

论文 [33] 对 autoencoder 进行了改进，结合 GAN 提出了新的 generative model。第一次看了一两页，看的很懵，第二次看的时候十分痛快，一个上午就看完了，整篇论文读下来收获很多。Adversarial Autoencoder (AAE) 将 GAN 与 autoencoder 相结合，仿照 GAN 的训练方式来训练自编码器网络。通篇看下来，autoencoder、GAN、AAE 都成了分布的匹配（拟合）问题。特别是论文提出的 AAE，论文将数据分布  $p_d(\mathbf{x})$  的隐变量  $\mathbf{x}$  的分布与某个已知的分布进行匹配/拟合。论文中不仅给出了基础的 AAE 的架构，如图 Fig.12 所示，还给出了 AAE 在其他方面的应用，如有/无/半监督的 AAE、AAE 在聚类和降维上的应用，并给出了相应的网络结构，非常棒！

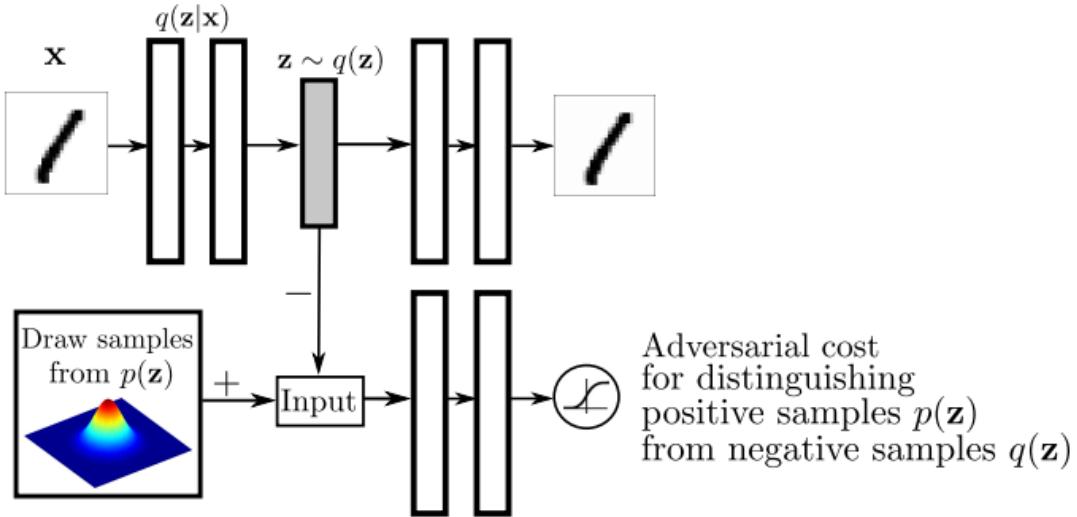


Figure 12: Architecture of AAE

**AAE 思路** 通常的 autoencoder 或者 VAE[25] 在训练时通常是以重建损失或者 ELBO 为目标，VAE 中使用了重参数化的技巧来训练。AAE 其实和 VAE 也是有一点相似之处的：都会从一个确定的分布中进行采样。对 AAE 而言，整个网络结构分为两部分：autoencoder 和 discriminator。autoencoder 就是普通的自编码器：输入数据学习到数据的隐表示  $z$ ，使用重建损失函数；discriminator 与 GAN 中的 D 时一致的：从一个已知的先验分布  $p(z)$  中采样一个  $z$ ，这个时候来自 autoencoder 的  $z$  作为负样本  $z_-$ ，来自  $p(z)$  作为正样本  $z_+$ ，由 D 来进行判别，使用判别损失函数。如果与 GAN 进行比较，可以知道  $p(z)$  相当于真是数据分布，而 autoencoder 中的 encoder 则相当于 GAN 中的 generator。上文中还提到了一个很关键的点：**分布的匹配/拟合**。

这在 AAE 中体现的尤为明显。上面描述的过程其实也可以抽象为分布的匹配/拟合。已知的分布  $p(z)$  作为一个目标分布，目的是使数据隐变量的分布  $p(z)$  与目的分布一致 — 判别器无法区分  $z$  是来自  $p(z)$  还是来自  $q(z)$ 。与 VAE 中的重参数化不同，AAE 可以直接使用梯度下降的方法进行训练。AAE 的训练过程分为两个阶段：reconstruction 和 regularization。

在 reconstruction 阶段：autoencoder 通过最小化重建损失来更新 encoder 和 decoder。在 regularization 阶段：通过对抗损失来更新判别网络和生成网络（也即 encoder）。

**AAE 的应用** 论文中还介绍了 AAE 的几个主要的应用：

- 将标签信息融入到判别器中：以 one-hot 的形式构造标签编码，将标签编码与从  $p(z)$  中采样到的  $z$  一起输入到判别网络中
- 将标签信息融入到 autoencoder 中：将 one-hot 编码的标签信息与数据的隐变量  $z$  一起输入到 decoder 中
- 半监督的 AAE：其实这个相当于将标签信息融入到 autoencoder 和判别器中。此时的网络结构中将会由两个判别器，分别对  $z$  和标签信息进行判别，因此也需要为标签信息增加一个 generator — 论文中采用的是

categorical 分布。此时网络的训练在原基础上增加了一个阶段 — semi-supervised classification 阶段，该阶段使用交叉熵来更新监督损失

- 使用 AAE 进行无监督的聚类：这部分十分有趣！论文中特地针对 MINST 数据集进行了聚类，但不是聚类成 10 类，AAE 可以将同一个数字的不同风格的图像区分出来 — 将风格分离了出来
- 使用 AAE 进行降维：此处的 AAE 与半监督的 AAE 十分相似。论文中还分析了已有方法进行降维时的几个主要问题：难以应用到新的数据上 (T\_SNE 解决了这个问题)、数据可能处于流形空间等

### 方法解决的问题/优势

- 将 GAN 中的 discriminator 与 autoencoder 相结合，得到了强大的自编码器
- 不需要使用重参数化就能训练，效果比 VAE 好
- AAE 的结构修改可以使用到多种任务上，可扩展性极好

### 方法的局限性/未来方向

- 用来学习更好的表征，很棒的特征提取器
- **基于 AAE 的聚类**
- **基于 AAE 的降维可视化**

## 11 A Comprehensive Survey on Graph Neural Network

论文地址：<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9046288>

来源：IEEE Transactions on Neural Networks and Learning Systems, 2020

作者：Zonghan Wu, Shirui Pan et.al

关键词：Deep Learning, graph neural networks, graph convolutional networks, graph representation learning, graph autoencoders

写于：2020-12-27

该论文 [60] 对今年来图神经网络的发展和应用进行了全面的总结，包括对最近的 GNNs 进行分类、GNNs 在不同领域内的应用、开源的代码和基准数据集、GNNs 的模型评估以及 GNN 未来可能的发展方向。

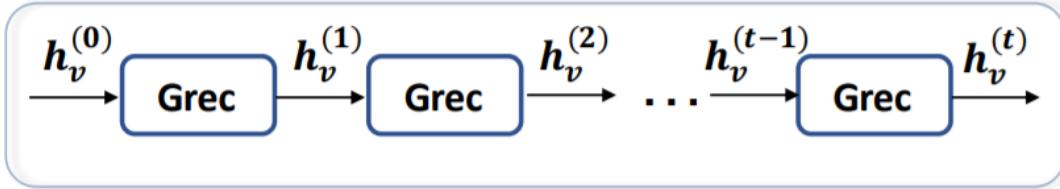
**GNNs 分类** 该论文将 GNNs 分为四大类：recurrent gnns RecGNNs）、convolutional gnns (ConvGNNs)、graph autoencoders (GAEs)、spatial-temporal gnns (STGNNs)。

**RecGNNs** 大多是 GNNs 较为早期的工作，目的是通过循环的神经结构来学习结点/图的表征，结点间一轮一轮地交换信息就相当于循环地过程。[28] 就是其中的一种。RecGNN 的结点表征更新如下：

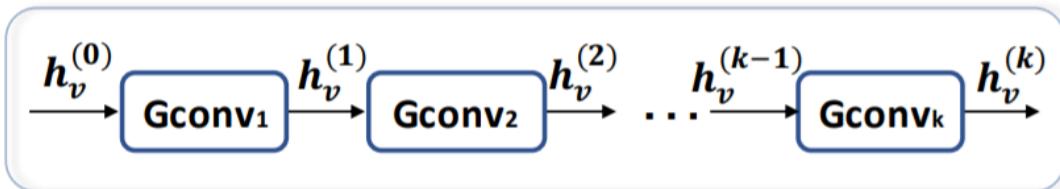
$$\mathbf{h}_v^{(t)} = \sum_{u \in N(v)} f(\mathbf{X}_v, \mathbf{X}_{(v,u)}^e, \mathbf{h}_u^{(t-1)})$$

其中  $f$  是 recurrent function，该函数应该是一个压缩映射（可以用 NN 来实现），能够将输入映射到一个更紧凑的空间中。本质上，RecGNNs 是基于信息传播的模型的，通过一轮轮的传播来更新表征。

**ConvGNNs** 现在最常见的 GNNs — 图卷积神经网络。ConvGNNs 和 RecGNNs 很像，但是有一个很明显的区别 — ConvGNNs 通过不同的层来完成结点之间信息的交换，且层所代表的映射可以不是压缩映射，且 ConvGNNs 的层一般都是不一样的，而 RecGNNs 中的  $f$  通常都是同一个函数。二者的对比如 Fig.13 所示。



(a) Recurrent Graph Neural Networks (RecGNNs). RecGNNs use the same graph recurrent layer (Grec) in updating node representations.



(b) Convolutional Graph Neural Networks (ConvGNNs). ConvGNNs use a different graph convolutional layer (Gconv) in updating node representations.

Figure 13: RecGNNs v.s. ConvGNNs

在这四种 GNNs 中，ConvGNNs 算是发展得最迅猛、相关工作最多的一类了。ConvGNNs 又可以分为两类：Spectral-based ConvGNNs 和 Spatial-based ConvGNNs。

**Spectral-based** 的方法从图信号处理的角度定义了图卷积。Spectral-based 默认 graph 是无向的，spectral 下的图卷积需要对图的 Laplacian 矩阵进行谱分解  $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T$ ，对 graph signal 进行卷积需要进行 *graph Fourier transform*，即  $F(\mathbf{x}) = \mathbf{U}^T \mathbf{x}$ ，其中  $\mathbf{x}$  为输入的 graph signal。图傅里叶变换能够将 graph signal 变换到由  $\mathbf{U}$  定义的正交空间中。那么给定一个 filter (和图像处理领域的滤波器很像)  $\mathbf{g}$ ，那么对一个 graph signal 的卷积定义如下：

$$\begin{aligned} \mathbf{x} *_{\mathcal{G}} \mathbf{g} &= F^{-1}(F(\mathbf{x}) \odot F(\mathbf{g})) \\ &= \mathbf{U} (\mathbf{U}^T \mathbf{x} \odot \mathbf{U}^T \mathbf{g}) \end{aligned}$$

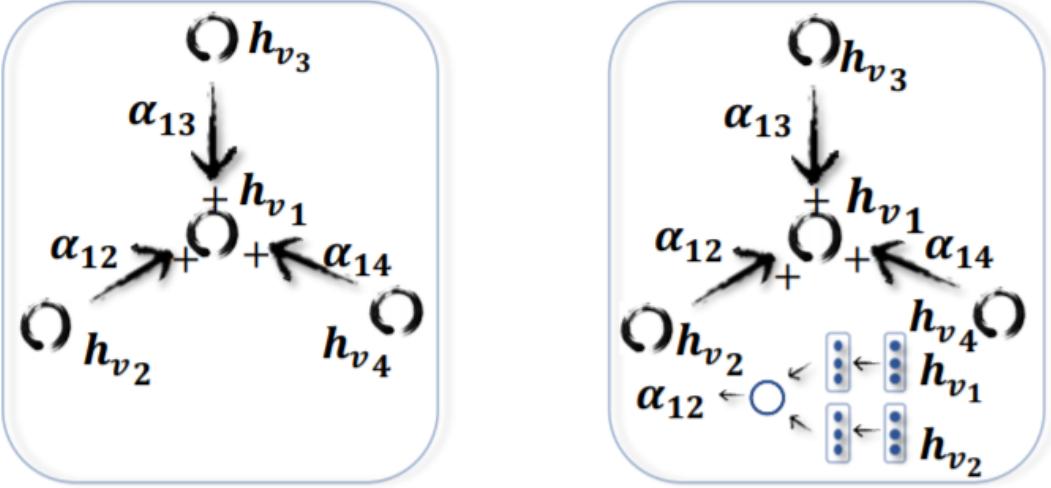
那么 Spectral CNN 的定义如下，其中  $\Theta_{i,j}^{(k)}$  是需要学习的滤波器：

$$\mathbf{H}_{:,j}^{(k)} = \sigma \left( \sum_{i=1}^{f_{k-1}} \mathbf{U} \Theta_{i,j}^{(k)} \mathbf{U}^T \mathbf{H}_{:,i}^{(k-1)} \right) \quad (j = 1, 2, \dots, f_k)$$

spectral-based 的图卷积有很坚实的数学基础，但也存在一些局限性：

- 不同的结点排列会产生不同的谱分解 — 不是结点排列无关的
- 学习到的滤波器是领域相关的 — 难以应用到其他领域
- 矩阵分解计算复杂 — 难以应用到 large scale graph 上

**Spatial-based** 即空域的图卷积，是目前的主流。空域的图卷积与图像中的 CNN 很像。过程：将邻居结点的信息汇聚后，再通过某个函数将结点自身的信息与汇聚后的邻居信息结合起来，与 spectral-based 的方法相比更直观。在空域卷积的基础上还诞生了图注意力网络 Graph Attention Network。与基础的空域卷积相比，结点与其邻居结点间的权重改由注意力替代。二者的比较如 Fig.14 所示：



(a) GCN [22] explicitly assigns a non-parametric weight  $a_{ij} = \frac{1}{\sqrt{\deg(v_i)\deg(v_j)}}$  to the neighbor  $v_j$  of  $v_i$  during the aggregation process.

(b) GAT [43] implicitly captures the weight  $a_{ij}$  via an end-to-end neural network architecture, so that more important nodes receive larger weights.

Figure 14: Difference between GCN and GAT

除了 GAT 还出现了与图像领域中类似的操作，如 Graph Pooling，功能也与图像领域的池化操作类似 — 降低图的规模/从结点表征矩阵产生图表征 (Readout)。

**GAEs** 图自编码器 — encoder 将 graph 的结点映射到隐空间中，decoder 将隐空间中的图还原/重建。GAEs 可以用来学习结点的表征以及生成图。一个现有的 GAEs 的一个统计表如 Fig.15 所示。使用 GAEs 来生

| Approaches           | Inputs      | Encoder                  | Decoder                  | Objective  |
|----------------------|-------------|--------------------------|--------------------------|--|
| DNGR (2016) [59]     | $A$         | a multi-layer perceptron | a multi-layer perceptron | reconstruct the PPMI matrix                                |
| SDNE (2016) [60]     | $A$         | a multi-layer perceptron | a multi-layer perceptron | preserve node 1st-order and 2nd-order proximity            |
| GAE* (2016) [61]     | $A, X$      | a ConvGNN                | a similarity measure     | reconstruct the adjacency matrix                           |
| VGAE (2016) [61]     | $A, X$      | a ConvGNN                | a similarity measure     | learn the generative distribution of data                  |
| ARVGA (2018) [62]    | $A, X$      | a ConvGNN                | a similarity measure     | learn the generative distribution of data adversarially    |
| DNRE (2018) [63]     | $A$         | an LSTM network          | an identity function     | recover network embedding                                  |
| NetRA (2018) [64]    | $A$         | an LSTM network          | an LSTM network          | recover network embedding with adversarial training        |
| DeepGMG (2018) [65]  | $A, X, X^e$ | a RecGNN                 | a decision process       | maximize the expected joint log-likelihood                 |
| GraphRNN (2018) [66] | $A$         | a RNN                    | a decision process       | maximize the likelihood of permutations                    |
| GraphVAE (2018) [67] | $A, X, X^e$ | a ConvGNN                | a multi-layer perceptron | optimize the reconstruction loss                           |
| RGVAE (2018) [68]    | $A, X, X^e$ | a CNN                    | a deconvolutional net    | optimize the reconstruction loss with validity constraints |
| MolGAN (2018) [69]   | $A, X, X^e$ | a ConvGNN                | a multi-layer perceptron | optimize the generative adversarial loss and the RL loss   |
| NetGAN (2018) [70]   | $A$         | an LSTM network          | an LSTM network          | optimize the generative adversarial loss                   |

Figure 15: Main characteristics of selected GAEs

成图可以分为两类：Sequential — 逐步地增减结点/边，Global — 一次性生成整个 graph。

**STGNNs** 时空地图神经网络，这种 GNNs 主要用在 dynamic graphs 上 — 捕捉时间序列地特征/分布，序列与 graph 的联合/边缘分布，graph 在时间、空间上的关系。STGNNs 的任务可以是预测预测结点的值/标签或者 graph 的便签 — 当然是结合时间的。常见的应用场景包括流式的图数据、交通数据（如交通中的流量、速度等数据）等。STGNNs 目前主要由两个发展方向：RNN-based 和 CNN-based。

RNN-based 方法会单独处理每一个时刻的图数据，并作为输入流入下一个时刻的处理步骤。有些方法会将结点和边分开来处理 — 分别输入 RNN-based 处理单元中。CNN-based 方法以非循环的方式来处理，RNN-based 和 CNN-based 有点像 RecGNNs 和 ConvGNNs，CNN-based 方法通过堆叠多层网络来处理 dynamic graph。

## GNNs 应用

**Computer Vision** 包括场景图生成、点云数据分类、动作识别。在场景图中，GNNs 可以帮助识别对象之间的语义关系，与之相反的是场景图的生成，将自然语言描述的场景生成为场景图。

**Natural Language Processing** GNNs 可以借助文本中句子/词之间的关系来对文本进行分类，除此之外，在 NLP 领域还有很多自然的 grpah-like 数据，如语法依存树。同时也可以基于语义图 (semantic graph) 来生成文本或者反过来。

**Traffic** 例如动态预测道路的可行驶速度、流量等，这通常与 STGNNs 联系起来。通常将交通网络看作时空图，其中的结点为安装在路上的传感器、边表示传感器之间的距离，这样每个结点在一定的时间区间内都有自己的流速。

**Recommender system** Graph-based 的推荐系统通常将用户、商品视作结点，通过利用结点之间的关系以及与相关的信息融合起来进行推荐。推荐系统的一个关键点：针对一个用户，给出每个商品对其的重要性排序/分数，这也同样可以转化为一个链接预测问题 — 预测用户和商品之间的链接。

**Chemistry** 在生物/化学/生物信息领域，通常利用 GNNs 来研究分子/化合物/药物的结构 — 这些东西也天然地具有 graph 结构。

**Others** 除了上述的几个主要的应用领域，还存在一些比较小众/还未广泛研究的领域，例如程序验证、程序推理、社会影响力预测、对抗攻击防御、异常检测、脑网络研究等。

**GNNs 模型评估** 在 GNNs 中有几个主要的任务：结点分类、图分类、链接预测（我自己补充的）。

**常用数据集** 常见的数据如 Fig.16 所示。

## 未来发展方向

**Model Depth** 就目前的情况来看，GNNs 的模型都比较浅，与 CNN 模型动则数十层的模型相比，GNNs 的模型层数都比较少。当 GNN 中层数较多时，基本上每个结点的信息都已经被传播到了图中其余的结点 — 通常图的最短路径会在 6 跳左右/六度分隔，这也导致当层数一多，图中结点的表征就会趋于一致，这反而使得结点的表征不具有区分性。**如何加深 GNNs 或者什么情况下需要深度的 GNNs 是一个很值得研究的问题**。

**Scalability trade-off** 伸缩性权衡。GNNs 的可伸缩性是以破坏图的完整性为代价的。在现有的 GNNs 中，通常会有 ZPooling 和 Coarsening 操作来降低节点特征的维度或者图的规模，但是这些操作都或多或少的丢失了一部分信息。如何设计一个破坏图数据完整性的 GNN 或者如何权衡完整性和可行性是一个很值得研究的问题。

| Category            | Data set    | Source | # Graphs | # Nodes(Avg.) | # Edges (Avg.) | #Features | # Classes |
|---------------------|-------------|--------|----------|---------------|----------------|-----------|-----------|
| Citation Networks   | Cora        | [117]  | 1        | 2708          | 5429           | 1433      | 7         |
|                     | Citeseer    | [117]  | 1        | 3327          | 4732           | 3703      | 6         |
|                     | Pubmed      | [117]  | 1        | 19717         | 44338          | 500       | 3         |
|                     | DBLP (v11)  | [118]  | 1        | 4107340       | 36624464       | -         | -         |
| Bio-chemical Graphs | PPI         | [119]  | 24       | 56944         | 818716         | 50        | 121       |
|                     | NCI-1       | [120]  | 4110     | 29.87         | 32.30          | 37        | 2         |
|                     | MUTAG       | [121]  | 188      | 17.93         | 19.79          | 7         | 2         |
|                     | D&D         | [122]  | 1178     | 284.31        | 715.65         | 82        | 2         |
|                     | PROTEIN     | [123]  | 1113     | 39.06         | 72.81          | 4         | 2         |
|                     | PTC         | [124]  | 344      | 25.5          | -              | 19        | 2         |
|                     | QM9         | [125]  | 133885   | -             | -              | -         | -         |
| Social Networks     | Alchemy     | [126]  | 119487   | -             | -              | -         | -         |
|                     | Reddit      | [42]   | 1        | 232965        | 11606919       | 602       | 41        |
| Others              | BlogCatalog | [127]  | 1        | 10312         | 333983         | -         | 39        |
|                     | MNIST       | [128]  | 70000    | 784           | -              | 1         | 10        |
|                     | METR-LA     | [129]  | 1        | 207           | 1515           | 2         | -         |
|                     | Nell        | [130]  | 1        | 65755         | 266144         | 61278     | 210       |

Figure 16: Summary of benchmark datasets

**Heterogeneity** GNNs 在异质图数据上的应用。目前大多数的 GNNs 处理的是同质的图数据 — 结点都为同一类型或者只有一种类型，边也只有一种类型，但是还有很多数据是异质的 — 结点/边的类型是多样的，以及结点/边的输入形式都是不一样的，例如结点/边为文本/图像等，这可以视作现在的多模态融合问题，相当于以 GNNs 来进行多模态数据融合。

**Dynamicity** 虽然已经有 STGNNs 来处理随时间变化的图数据，但是大部分都没有 dynamic spatioal relations (这里我不太理解)。不过确实需要设计更好的处理动态的图数据、流式的图数据 (流式的数据处理是一个很有意思的问题)。

这篇综述是 2019 年发表的，虽然在 2020 年已经出现了很多新的关于 GNNs 的工作，一些工作也与文中谈的一些问题重叠了，但是这篇综述依然是一片很全面、详细的对 GNNs 今年来的工作的总结，提出的问题也很具有参考性，是一篇非常棒的综述！

## 12 The Emerging Field of Signal Processing on Graphs

论文地址：<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6494675>

来源：IEEE Signal Processing Magazine, 2013

作者：David I Shuman, Sunil K. Narang, Pascal Frossard et.al

关键词：graph signal, signal processing, network analysis

写于：2021-01-15

该论文 [52] 写于 2013 年，对当时兴起的图信号处理领域进行了总结，主要的内容包括：领域内的主要挑战、定义图谱 (graph spectral) 领域的不同方法、处理图信号时应用图的不规则结构的重要性、将一些信号处理领域的操作推广到图信号处理领域。

怎么写这篇论文的笔记呢？其实这篇论文我也看的不是很懂，里面涉及很多信号处理领域的理论，也需要较强的数学功底，特别是很多操作是在傅里叶变换的基础上定义的，具体的理论部分我只是浅尝辄止。

论文中首先介绍 graph 形式的数据在生活中的广泛，当然现在已经有更多的 graph 形式的数据被发现。既然是图上的信号处理，那能不能将现有的信号处理的方法应用到图信号上呢？当然是可以的，但是也不能生搬硬套，**如何将现有的信号处理理论方法迁移到图信号上** 就是一个挑战。

紧接着论文介绍了谱图领域。图信号可以用  $\mathbb{R}^N$  中的一个点来表示（n 为图中结点数），并用图 Laplacian 矩阵定义了图信号的傅里叶变换。图信号可以在顶点域（vertex domain）和谱域（spectral domain）中定义，先前的定义是在顶点域上的定义。

最重要的部分就是将传统信号处理领域的理论和方法推广到图信号上了。推广的对信号的操作包括滤波（频率域和顶点域）、卷积、平移、调制和膨胀、图的粗化和下采样、readout 等。

一些领域内开放的问题：

- 如何使用 Laplacian
- 图中结点距离的度量方式很多，该使用哪种
- 对于大图，很多矩阵分解的方法不再适用
- 如何将图信号的性质与图的性质联系起来

## 13 Rethinking pooling in graph neural networks

论文地址：<https://arxiv.org/pdf/2010.11418.pdf>

来源：NeurIPS, 2020

作者：Diego Mesquita, Amauri H. Souza, Samuel Kaski

关键词：GNN, Pooling

写于：2021-01-17

该论文 [37] 对 GNN 中常用的 pooling 操作进行了思考，pooling 对 GNN 的性能有多大帮助？可谓是对很多论文中提出的各式各样的 pooling 打脸了啊。该论文选择了三种比较具有代表性的 pooling 进行研究，使用随机的操作替换原来的 pooling，结果性能基本没有变化（**尴尬**），之后作者分析了卷积与 pooling 之间的关系，解释了 pooling 不起作用的原因。

**问题定义** GNNs 中的 pooling 与 CNN 中的 pooling 类似，一般都是为了下采样，减小输入的规模。GNNs 中的 pooling 通过将 graph 中的结点聚合生成一个规模更小的 graph — 具有更少的结点数，结点特征也更少。大部分的 GNNs 中都用到了 pooling（有些论文中也叫 coarsening），也有一些论文专门提出了一些 pooling 操作。其实，graph 的 pooling 可以视作一个对结点进行聚类的过程，给每个结点赋予一个类别，将同一类别的结点视作 pooling 后的 graph 中的一个结点，节点特征矩阵和邻接矩阵也会更具 pooling 结果进行调整。具体的 pooling 可以根据结点的特征进行聚类（通常做法），也可以用神经网络来学习学习聚类的结果。

既然大家都用 pooling，它真的那么重要吗？

**Rethinking 思路** 作者选择了 3 个具有代表性的模型：GRACLUS[13]、DIFFPOOL[65]、GMN[23]。这 3 个模型里都使用了 local pooling，作者对其中的 pooling 操作进行了修改，例如替换为随机的 pooling、使用原 graph 的补 graph 进行 pooling 作为原 graph 的 pooling 结果等，在常用的数据集上对比了替换前后模型的效果。惊人的事情出现了一—模型的效果基本没有变化，有些甚至不如替换后的效果好!!!

在分析阶段，作者主要分析了卷积和 pooling 对模型的影响。作者通过实验说明，卷积层学到的结点特征是趋同的（同质的），而 pooling 则加重了这一现象，pooling 可能会使得结点特征的方差变小。一个卷积相当于

提取了一种特征，在实践中通常会使用多个卷积来提取特征。作者还实验了单个卷积和多个卷积对模型效果的影响 — 单个卷积在大多数情况下效果都会比多个卷积更差，但是当特征是结点特征是常量是多个卷积对效果并无很大影响。卷积的数量与初始时结点特征的丰富程度有关。

### 方法解决的问题/优势

- 很大胆的质疑了几乎所有 GNNs 中都用到了 pooling
- 在几个主流的 pooling 上进行了验证，与替换之后的 pooling 进行对比
- 分析了各种 pooling 效果甚微的原因，分析了卷积与 pooling 之间的联系

### 方法的局限性/未来方向

- pooling 也许并不是没用的。论文只是替换了 pooling，并没有取消 pooling，**或许具体的 pooling 没有那么重要，但不代表 pooling 不重要**
- 论文中所验证的数据集都是规模比较小的 graph，**当扩展到规模大得多的 graph 上时，或许这个时候才能体现 pooling 的作用**

## 14 FASTGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling

论文地址：<https://arxiv.org/pdf/1801.10247.pdf>

来源：ICLR, 2018

作者：Jie Chen, Tengfei Ma, Cao Xiao

源码：[FastGCN](#)

关键词：**GCN, GNN**

写于：2021-01-21

该论文 [7] 针对 GCN 训练和学习过程中地计算量大和耗时提出了 integral transform 的方法。本文所针对的 Graph 主要是稠密图和符合幂律的 Graph，当对其中一个结点卷积时，所覆盖的结点集可能已经很大了。同时，FastGCN 是 inductive 的。

本文的重点是对结点的采样以及从积分的角度来看待卷积层。

这篇论文没看太懂，可见博客：[源码分析-FastGCN](#)、[FastGCN](#)。

## 15 Weakly-Supervised Disentanglement Without Compromises

论文地址：<https://arxiv.org/pdf/2002.02886.pdf>

来源：ICML, 2020

作者：Francesco Locatello, Ben Poole

源码：[weak\\_disentangle](#)

关键词：**generative model, disentangled representation, weakly-supervised generative model**

写于：2021-01-22

该论文 [15] 是关于解耦表征学习的。论文提出的解耦表征学习方法从成对的图像中学习解耦表征，成对的图像之间存在一些区别，区别主要体现在隐因子的不同上。该方法只需要知道有几个隐因子变化了而不需要知道具体是哪几个。

**解耦表征** 目的是学习一个映射，该映射能够将数据映射到低维表示，表征是受潜在的因子控制的，每个潜在因子的变化能够使表征的一个或一些维度发生变化。这里所说的潜在因子与 VAE 中的隐变量很像。

我们可以假设任这世界上是有“终极”的，任何事物都是由一些因子/隐变量/终极特征所控制的。当我们用计算机来建模世界时，一些潜在的因子在控制表征。当进行表征学习时，我们用一个低维的向量来表示数据点，表征向量的某一（些）维发生变化对应着潜在因子的变化。有时我们学习的特征也学可以较好地应用于下游任务，但是它可能并不是解耦的。例如在 AE 中，我们可以学到重建误差很小的表征，但是隐变量与表征之间的对应关系可能并不那么准确，如控制颜色的因子并没有与表征中的颜色部分相对应。从相关的角度来看，表征的不同维度（或者维度子集间）之间没有尽可能地无关。为什么会这样呢？因为在重建误差地指导下，习得地表征并不一定与隐变量之间有很准确的一一对应关系，虽然重建效果好但是可能难以被我们理解，表征地不同维度之间是相关的。

解耦表征的目的就是解决上述问题。解耦指的是表征的某个（些）维度与潜在因子之间解耦，即一个因子控制一个（些）表征的维度，一个（些）表征的维度受一个因子的控制。这样做有什么好处呢？首先，学到的表征具有可解释性，对数据有更深层的理解；其次，可以控制表征的生成，知道表征与因子的对应关系可以实现可控的表征生成，相当于对表征空间有了更深入的了解；再者，对于下游任务，我们可以在解耦表征的基础上进行优化。

**问题定义** 其实，这篇论文呢我也不可能太懂，可能是里面的数学太多了。但是文章主要解决的问题是从成对儿的数据中学习到解耦表征。从论文的 title 可以看出，weakly-supervised。论文中的数据不需要知道有哪些因子，只需要知道有多少因子变化了即可，不需要对数据进行标注。

### Disentangled Representation 优点

- 解耦表征能够以压缩和可解释的结构包含原数据中的所有信息，并且表征与下游任务尽量无关
- 对下游的半监督或无监督学习任务有帮助，如迁移学习、少样本学习等
- 能够找出控制数据表征的重要 factors

## 16 Representation Learning: A Review and New Perspectives

论文地址：<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6472238>

来源：IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 2013

作者：Yoshua Bengio, Aaron Courville, and Pascal Vincent

关键词：deep learning, representation learning, feature learning

写于：2021-01-26

该论文 [6] 写于 2013 年，虽然有一点久远了，但全文内容十分充实、全面，不愧是 Bengio 巨佬的论文。该文主要对无监督的表征学习进行了总结，内容涵盖了概率模型、自编码器、流形学习以及神经网络。

机器学习的效果严重依赖于输入的数据 — 即特征工程，在机器学习相关的任务中大部分的时间其实都花在了处理数据、构建特征上。特征工程通常作为下游任务的输入，对于构建的特征，有一个很通用的标准：要有区分性。

## 表征学习的应用

语音识别与信号处理

目标识别

自然语言处理

多任务学习、迁移学习、领域适应

## 表征学习中的 Priors

**Smoothness** 输入空间中相近的数据点在表征空间中也应该是相近的。相近的概念要看各个空间中距离是如何定义的。这是最基本的一个 prior，但可能会导致 curse of dimensionality。通常输入空间中样本的分布是很复杂的，在表征空间中会对输入空间进行一定的压缩，这会导致在表征空间中产生一些 wrinkles — 可能有一些导数很大或者不可导的区域。为了在表征空间中尽量连续，则需要大量的数据来“抹平”这些 wrinkles。当输入空间的维数增大时，为了“抹平” wrinkles 所需的数据将呈指数形式增长 — curse of dimensionality。

**Multiple explanatory factors** 输入的数据可能是由一些潜在的因素生成的 — “有一双魔掌在控制着世间万物的生灭”。

**A hierarchical organization of explanatory factors** 潜在因素可以通过层次化的结构进行组织，即我们对数据的抽象是层次化的、逐步的。

**Semi-supervised learning** 当我们使用数据去预测目标时，控制输入数据的潜在因素同样也能控制目标。对  $P(X)$  有用的表征同样对  $P(Y|X)$  也是有用的。其实，我感觉这也可以从层次化的角度来看，目标可以看作输入的更深层次的抽象，输入和目标都是受潜在因素控制的，只不过二者的抽象程度不同。

**Shared factors across tasks** 不同的任务之间是会共享一些潜在因素的，这也是迁移学习有效果的原因之一。

**Manifold** 数据空间中，样本点并不是随机或者均匀分布的，而是呈现聚集的状态，数据会集中地占据空间的一部分。不同类型的数据之间数据点密度是较低的。

**Natural clustering** 同类的数据在输入/表征空间中应该是自然聚集的。

**Temporal and spatial coherence** 输入空间中的微小变化也对应着目标空间中的微小变化，与函数的连续性很相似。

**Sparsity** 给定一个样本点  $x$ ，只有一部分因素与之相关，具体来看，表征向量大部分维上取值为 0，既可以表示为表征向量对输入的微小变化是不敏感的。

**Simplicity of factor dependencies** 良好的表征，因素之间的关系应该是尽量简单的，如相互独立、线性关系等。

**What makes a Representation Good?** 首先，当然应该包含上述的 Priors。其次，还有以下因素：

#### Distributed representation

**Depth and Abstraction** 通常表征学习模型具有一定的深度 — 在学习数据表征的过程中是有一定层次的，能够帮助产生逐步抽象的数据。抽象的数据表征对一些变化是具有不变性的，如平移、旋转、缩放等。

**Disentangling Factors of Variation** 潜在因素之间尽可能使独立的。

**Good Criteria for Learning Representation?** 接下来论文对如何构建 deep representation 进行了介绍，比如通过逐层训练的方式来学习深度模型。

在表征学习中，主要可以分为两个流派：概率图模型、神经网络模型。可以简单的区分二者：隐藏单元是被视为隐变量还是可计算结点。接下来就是介绍各种模型了，其中概率模型一直没看懂，都是通过输入、表征的概率分布来建模的，一般是通过最大化似然概率来求解。

接下来是直接学习一个映射将数据从输入空间映射到表征空间，主要介绍了自编码器、regularized autoencoders。在去噪自编码器（从含噪声的样本中能够重建出无噪的样本）中，这需要借助一个假设：样本点使流形数据，数据点是聚集的。加入噪声后的数据会从一定程度上偏离流形，偏离到密度更低的区域。

其实，表征学习可以看作流形学习。有这么一个假设：真实世界中的数据所处的空间是非常大的，甚至是无限维的，数据可以用一个更低维的空间来表示。接下来论文就介绍了一些流形学习的方法。

接下来论文介绍了训练深度模型时的一些问题，如 1) 参数初始化的重要性；2) 参数稀疏性的重要性，当大部分参数一起更新时，目标函数可能会在多个方向上移动，这使得收敛变得困难；3) 数值计算的问题，如病态方程等；4) 标记数据对模型的效果的提升。

这篇论文真的是非常厚重了，内容足足的，可惜还有不少内容我看不太懂，哎，巨佬就是巨佬啊，非一日之功啊！

## 17 Contrastive and Generative Graph Convolutional Networks for Graph-based Semi-Supervised Learning

论文地址：<https://arxiv.org/pdf/2009.07111.pdf>

来源：AAAI, 2021

作者：Sheng Wan, Shirui Pan, Jian Yang, Chen Gong

关键词：semi-supervised learning, GCN, Contrastive Learning

写于：2021-03-01

该论文 [56] 主要解决的是基于图的半监督学习中的监督信息短缺的问题，该论文结合数据之间的相似性和图结构来丰富监督信息。

**问题定义** 假设共有  $n = l + u$  个样本（结点）， $\Psi = \{\mathbf{x}_1, \dots, \mathbf{x}_l, \mathbf{x}_{l+1}, \dots, \mathbf{x}_n\}$ ，其中前  $l$  个样本为带标签的样本，标签为  $\{y_i\}_{i=1}^l$ ，后  $u$  个样本不带标签，通常在半监督学习中  $l$  远小于  $u$ 。特征矩阵为  $\mathbf{X} \in \mathbb{R}^{n \times d}$ ， $\mathbf{Y} \in \mathbb{R}^{n \times c}$  为标签矩阵。该论文的目标就是找到后  $u$  个结点的标签。（这是一个 transductive 的方法）。

**思路** 整体框架如 Fig.17 所示。为了丰富监督信息，论文使用结点之间的相似性和图结构来丰富监督信息，分别使用半监督的对比学习来捕获数据之间的相似性和图生成损失来捕获图结构信息。

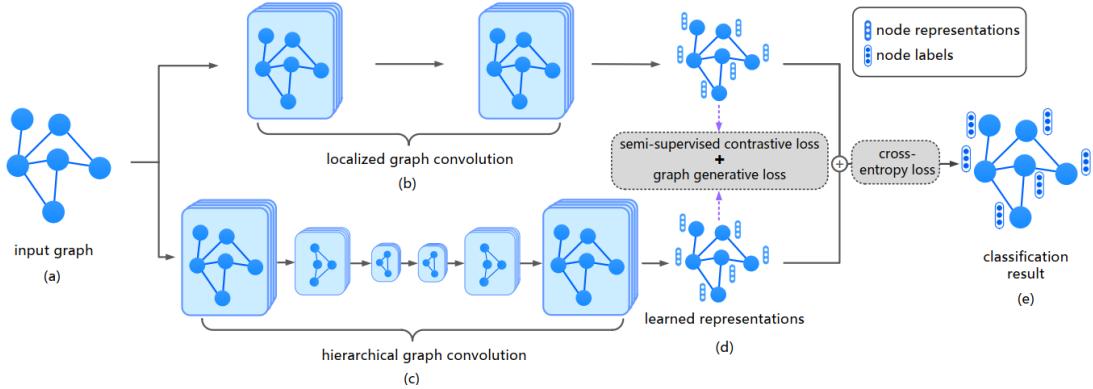


Figure 17: CG3

**Semi-Supervised Contrastive Learning** 对比学习通过数据之间的相似性与相异性来学习数据的表征，但是对比学习通常在无标签数据中，不能利用标签数据。为此，论文中提出半监督的对比学习来利用带标签的数据。半监督的对比学习损失可以分为两部分：无监督的对比损失和监督的对比损失。

论文中使用 local 和 global 两种视角的 GCN 来生成结点表征（目的是为了后续的对比学习），分别表示为  $\mathbf{H}^{\phi_1}, \mathbf{H}^{\phi_2}$ 。无监督的对比损失为：

$$\mathcal{L}_{uc} = \frac{1}{2n} \sum_{i=1}^n (\mathcal{L}_{uc}^{\phi_1}(\mathbf{x}_i) + \mathcal{L}_{uc}^{\phi_2}(\mathbf{x}_i))$$

其中  $\mathcal{L}_{uc}^{\phi_1}(\mathbf{x}_i), \mathcal{L}_{uc}^{\phi_2}(\mathbf{x}_i)$  分别为：

$$\begin{aligned}\mathcal{L}_{uc}^{\phi_1}(\mathbf{x}_i) &= -\log \frac{\exp(\langle \mathbf{h}_i^{\phi_1}, \mathbf{h}_i^{\phi_2} \rangle)}{\sum_{j=1}^n \exp(\langle \mathbf{h}_i^{\phi_1}, \mathbf{h}_j^{\phi_2} \rangle)} \\ \mathcal{L}_{uc}^{\phi_2}(\mathbf{x}_i) &= -\log \frac{\exp(\langle \mathbf{h}_i^{\phi_2}, \mathbf{h}_i^{\phi_1} \rangle)}{\sum_{j=1}^n \exp(\langle \mathbf{h}_i^{\phi_2}, \mathbf{h}_j^{\phi_1} \rangle)}\end{aligned}$$

有监督的对比损失为：

$$\mathcal{L}_{sc} = \frac{1}{2l} \sum_{i=1}^l (\mathcal{L}_{sc}^{\phi_1}(\mathbf{x}_i) + \mathcal{L}_{sc}^{\phi_2}(\mathbf{x}_i))$$

其中  $\mathcal{L}_{sc}^{\phi_1}(\mathbf{x}_i), \mathcal{L}_{sc}^{\phi_2}(\mathbf{x}_i)$  分别为：

$$\begin{aligned}\mathcal{L}_{sc}^{\phi_1}(\mathbf{x}_i) &= -\log \frac{\sum_{k=1}^l \mathbb{1}_{[y_i=y_k]} \exp(\langle \mathbf{h}_i^{\phi_1}, \mathbf{h}_k^{\phi_2} \rangle)}{\sum_{j=1}^l \exp(\langle \mathbf{h}_i^{\phi_1}, \mathbf{h}_j^{\phi_2} \rangle)} \\ \mathcal{L}_{sc}^{\phi_2}(\mathbf{x}_i) &= -\log \frac{\sum_{k=1}^l \mathbb{1}_{[y_i=y_k]} \exp(\langle \mathbf{h}_i^{\phi_2}, \mathbf{h}_k^{\phi_1} \rangle)}{\sum_{j=1}^l \exp(\langle \mathbf{h}_i^{\phi_2}, \mathbf{h}_j^{\phi_1} \rangle)},\end{aligned}$$

半监督对比损失为： $\mathcal{L}_{ssc} = \mathcal{L}_{uc} + \mathcal{L}_{sc}$ 。

**Graph Generative Loss** 为了利用图的结构作为监督信息引入了图生成损失。在现有生成模型的启示下，将图中边  $e_{ij}$  视为二元变量，并且该变量是条件独立的。所以在给定 local 和 global 视角的结点表征时，图的概率表示为：

$$\begin{aligned}p(\mathcal{G}|\mathbf{H}^{\phi_1}, \mathbf{H}^{\phi_2}) &= \prod_{i,j} p(e_{ij}|\mathbf{H}^{\phi_1}, \mathbf{H}^{\phi_2}) \\ &= \prod_{i,j} p(e_{ij}|\mathbf{h}_i^{\phi_1}, \mathbf{h}_j^{\phi_2}) \\ &= \prod_{i,j} \delta([\mathbf{h}_i^{\phi_1}, \mathbf{h}_j^{\phi_2}] \mathbf{w})\end{aligned}$$

(类似于极大似然概率)，其中  $\delta$  为逻辑回归函数。

图生成损失为： $\mathcal{L}_{g^2} = -p(\mathcal{G}|\mathbf{H}^{\phi_1}, \mathbf{H}^{\phi_2})$ 。

**Model Training** 因为采用了 local 和 global 的视角, 结点最终的表征为:  $\mathbf{O} = f\lambda^{\phi_1}\mathbf{H}^{\phi_1} + (1 - \lambda^{\phi_1})\mathbf{H}^{\phi_2}$ 。因为还存在一部分带标签的数据, 因此可以借助这一部分数据产生交叉熵损失:  $\mathcal{L}_{ce} = -\sum_{i=1}^l \sum_{j=1}^c \mathbf{Y}_{ij} \ln \mathbf{O}_{ij}$ 。最终模型的损失即为:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_{ssc} \mathcal{L}_{ssc} + \lambda_{g^2} \mathcal{L}_{g^2}$$

其中  $\lambda_{ssc}, \lambda_{g^2}$  均为超参数。

### 方法解决的问题/优势

- 将对比学习引入到半监督学习中, 结合了不带标签的数据和带标签的数据
- 利用结点相似性与图结构来丰富监督信息

### 方法的局限性/未来方向

- 论文中的方法为 transductive, 不能应用于未见过的结点
- **将论文中的方法改为 inductive 的**

## 18 Inductive and Unsupervised Representation Learning on Graph Structured Objects

论文地址: <https://openreview.net/pdf?id=rkem91rtDB>

来源: ICLR, 2020

作者: Lichen Wang, Bo Zong, et al.

源码: [SEED-Reimplementation](#)

slides: [SEED](#)

关键词: **unsupervised learning, graph representation**

写于: 2021-03-01

该论文 [57] 主要解决的是图结构数据的无监督的、inductive 形式的表征问题。通常在无监督的图表征问题中, 主要以重建损失为主导进行训练, 但是在计算重建损失时通常要涉及到图的相似性计算, 而图的相似性计算是一个十分复杂、耗时的过程, 论文提出了一个通用的框架 SEED (Sampling, Encoding and Embedding Distribution) 用于无监督的学习图结构对象的表征。

**问题定义** 目标很简单, 给定一个 graph, 学习它的表征。

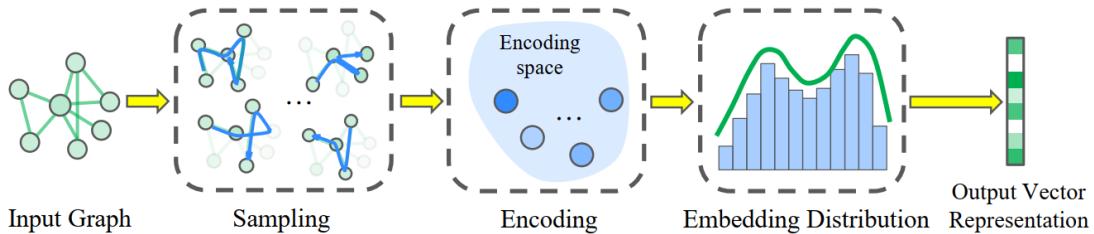


Figure 18: SEED

**SEED 思路** 如 Fig.18 所示, SEED 主要分为三个部分:

**Sampling** 从输入的图中采样出多个子图。为了使得采样到的子图更具代表性，论文中提出了一种新的采样方法 — WEAVE (random Walk with Earliest Visit time)。该方法与通常的随机游走不一样，WEAVE 是带结点访问时间戳的。如 Fig.19 所示，WEAVE 的区分能力比平凡的搜集游走更强。每一个 WEAVE 都代表一个

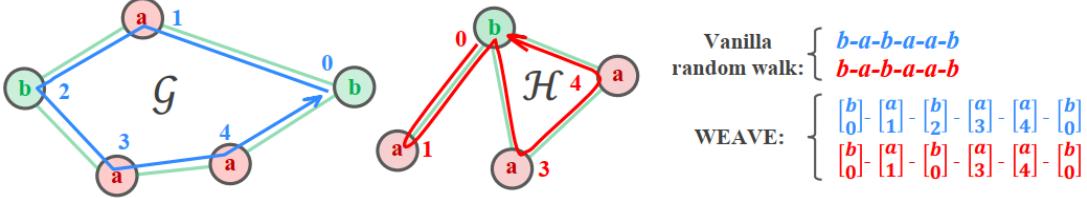


Figure 19: WEAVE 与随机游走对比

采样到的子图，可以用一个矩阵表示： $X = [\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)}]$ ，其中  $\mathbf{x}^{(p)} = [\mathbf{x}_a^{(p)}, \mathbf{x}_t^{(p)}]$ ， $\mathbf{x}_a^{(p)}$  表示在时间  $p$  时访问到的结点的特征， $\mathbf{x}_t^{(p)}$  表示访问到该结点时的时间向量。**注意，如果访问到了已经访问过的结点则  $\mathbf{x}_t^{(p)}$  为最早访问时的时间**。在论文中， $\mathbf{x}_t^{(p)}$  采用 one-hot 编码。

**Encoding** 将每一个采样到的子图编码为向量。直觉上，如果子图的表征质量好，那么就能在子图表征基础上较好地重建子图。论文中作者采样自编码器学习子图的表征，以重建损失作为损失函数。至此， $s$  个子图  $\{X_1, \dots, X_s\}$  被表示为  $s$  个向量  $\{\mathbf{z}_1, \dots, \mathbf{z}_s\}$ 。

**Embedding Distribution** 将上一阶段获得的多个子图的表征汇集作为输入图的表征。对于两个图，它们在表征空间中的距离应该与它们的子图向量分布距离类似，因此需要找到一个好的聚集函数来保留原先的子图表征分布距离，论文中采用的是  $MMD$ 。给定连个图  $\mathcal{G}, \mathcal{H}$ ，子图表征分别为： $\{\mathbf{z}_1, \dots, \mathbf{z}_s\}$  和  $\{\mathbf{h}_1, \dots, \mathbf{h}_s\}$ ，则两者间的  $MMD$  为：

$$\begin{aligned} \widehat{MMD}(P_{\mathcal{G}}, P_{\mathcal{H}}) &= \frac{1}{s(s-1)} \sum_{i=1}^s \sum_{j \neq i}^s k(\mathbf{z}_i, \mathbf{z}_j) + \frac{1}{s(s-1)} \sum_{i=1}^s \sum_{j \neq i}^s k(\mathbf{h}_i, \mathbf{h}_j) \\ &\quad - \frac{2}{s^2} \sum_{i=1}^s \sum_{j=1}^s k(\mathbf{z}_i, \mathbf{h}_j) \\ &= \|\hat{\mu}_{\mathcal{G}} - \hat{\mu}_{\mathcal{H}}\|_2^2 \end{aligned}$$

$\hat{\mu}_{\mathcal{G}}, \hat{\mu}_{\mathcal{H}}$  分别表示两个图的 kernel embedding，也就是最终的 graph representation，分别定义为：

$$\hat{\mu}_{\mathcal{G}} = \frac{1}{s} \sum_{i=1}^s \phi(\mathbf{z}_i), \quad \hat{\mu}_{\mathcal{H}} = \frac{1}{s} \sum_{i=1}^s \phi(\mathbf{h}_i)$$

其中  $\phi(\cdot)$  是与核函数  $k(\cdot, \cdot)$  相关的特征映射函数（与 SVM 中的核技巧类似，将核函数的计算转化为更简单的计算形式）。根据核函数的选择， $\phi(\cdot)$  具有不同的形式，如 RBF、MLP 等。为了训练  $\phi(\cdot)$ ，文中使用如下的近似误差，其中  $\theta_m$  为  $\phi(\cdot)$  的参数）：

$$J(\theta_m) = \left\| D(P_{\mathcal{G}}, P_{\mathcal{H}}) - \widehat{MMD}(P_{\mathcal{G}}, P_{\mathcal{H}}) \right\|_2^2$$

通过最小化上述误差，就能学习到较好的聚集函数，在最终的表征中保留子图表征的分布距离。

该论文的方法与核方法有一定的相似性。论文还证明了同构的图的 WEAVE 的子图分布是类似的，并且对子图的采样长度进行了证明，详细内容可以参考论文。

## 方法解决的问题/优势

- 给出了无监督形式的、inductive 的图结构对象表征学习方法

- 避免了复杂的图相似性计算，以类似于核技巧的方法较好地度量了图之间地距离
- 对相关地定理进行了证明

## 方法的局限性/未来方向

- 当图地规模较大时，采样的子图也会非常大，且可能需要采样地子图数量会很大

## 19 Attention is All you need

论文地址：<https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>

来源：NIPS, 2017

作者：Ashish Vaswani, et al.

源码：[tensor2tensor](#)

关键词：Sequence Modeling, Neural Machine Translation

写于：2021-03-07

该论文 [55] 针对序列转换(Sequence Transduction)问题中常见的弊端，提出使用 Attention 来完成 Sequence Transduction，提出了新的网络结构 — Transformer。

**问题定义** 在 Sequence Transduction 领域中，通常是基于 encoder-decoder 形式的循环神经网络或 CNN 的。也存在一些模型使用注意力机制将 encoder 和 decoder 连接起来。该论文提出了一种完全基于注意力机制的新型网络 — Transformer，不需要 RNN 或 CNN。

在对序列进行处理时，RNN 通常沿着输入/输出序列的顺序进行计算，将序列中元素的位置与计算的步骤相对齐。这样的处理方式看似天然地保留了序列形式，但是使得计算只能串行进行，难以并行，而且对于长序列的计算还会带来内存容量限制的问题。虽然目前已经有一些相关的工作关于 RNN 的并行化，但是 RNN 串行计算的问题依然存在。

注意力机制已经成为了序列建模中不可或缺的一部分，在建模序列中依赖的时候能够不用考虑输入/输出中的距离。但是之前的工作都是用于起连接作用，本文则将 encoder-decoder 完全建立在注意力机制上来完成序列的建模问题。并在机器翻译任务中检测了 Transformer 的效果。

### Transformer

**Encoder and Decoder** Transformer 的结构如 Fig.20 所示，其中左边为  $N$  个相同的 layer 堆叠起来的组成的 encoder，右边为  $N$  个相同的 layer 堆叠起来组成的 decoder。组成 encoder 的 layer 又两部分组成：Multi-head Attention、Fully connected Feed Forward。组成 encoder 的 layer 由三部分组成：两个 Multi-head Attention、Fully connected Feed Forward。以上的 Multi-head Attention 和 Fully connected FFN 都有残差连接和 layer normalization。这样每一部分的输出都可以表示为： $\text{LayerNorm}(x + \text{Sublayer}(x))$ ，其中 Sublayer 就是 Multi-head Attention 或 Fully connected FFN。

其实 Encoder 和 Decoder 都是由 Multi-head Attention 和 Fully connected FFN 组成的，但是二者在 Self-Attention 上有一些差别。Decoder 修改了 Self-Attention，是为了保证在预测  $i$  处的输出时，只能依赖  $i$  之前的输出，同时也是为了保持自回归（auto-regressive）的性质。

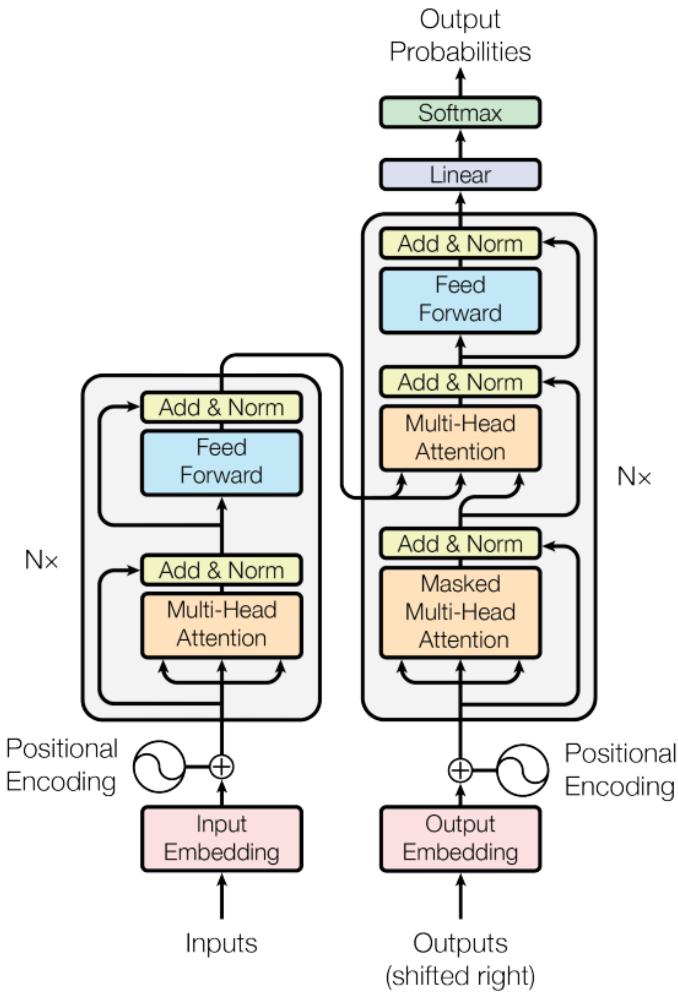


Figure 20: Transformer Architecture

**Attention** 一个注意力函数可以描述为：将一个 query 与一系列的 key-value 对（pair）进行映射得到一个输出，其中的 query、keys、values 和输出均为向量。输出时 values 的加权求和，values 对应的权是通过 query 与 key 计算得到的。

该论文中采用的是 **Scaled Dot-Product Attention**，其计算过程如 Fig.21 所示，写成公式为：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

有两种常用的 attention 函数：additive attention 和 dot-product attention。该论文中使用的是 dot-product attention，但是略微不同，是 Scaled 后的 dot-product attention。**为什么要对 dot-product 进行 scaled 呢？**当 query、key 的维度很大时，点积后的值可能会很大，可能会使 softmax 函数进入梯度很小的区域。举个例子，加入 q 和 k 都是服从均值为 0，方差为 1 的分布的随机变量，那么  $q \cdot k = \sum_{i=1}^d q_i k_i$ ，结果的均值为 0，但是方差为  $d$ 。

论文中使用了 **Multi-head Attention**。从 Fig.21 可以看出，在进行 Multi-head Attention 之前，先对 Q、K、V 进行了线性映射：

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat} (\text{head}_1, \dots, \text{head}_h) W^O \\ \text{where } \text{head}_i &= \text{Attention} (QW_i^Q, KW_i^K, VW_i^V) \end{aligned}$$

在 Multi-Head Attention 后就是 **Position-wise Feed-Forward Networks**。从名字 — Position-wise 也可以看出，Multi-head Attention 的输出是逐个通过 FFN 的。**在输入 Attention 函数之前，各个位置上的元素**

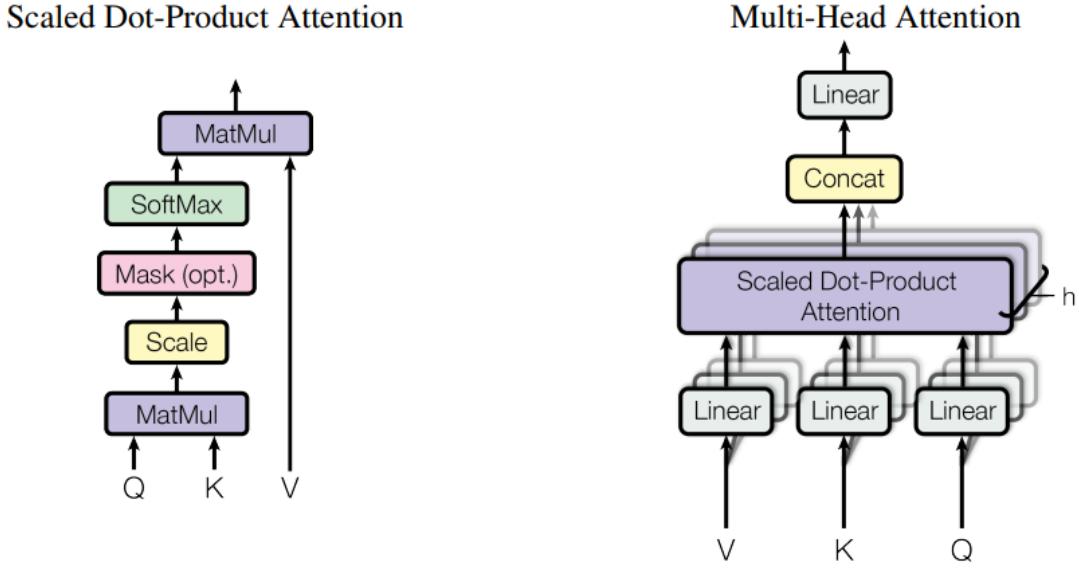


Figure 21: Scaled Dot-Production Attention 与 Multi-Head Attention

可以看作是独立的，不包含元素之间的依赖关系，经过 Attention 函数后，各个元素不仅包含了本身的含义还包含了与其他元素的依赖关系。所以不需要串行地通过 FFN，FFN 这一步操作是可以并行化的。

因为 Transformer 本身是不包含循环和卷积的，为了使模型利用序列中的顺序，论文中将 **Positional Encoding** “加” 到输入序列中的元素上。论文中使用的 Positional-Encoding：

$$PE_{(pos,2i)} = \sin \left( pos / 10000^{2i/d_{\text{model}}} \right)$$

$$PE_{(pos,2i+1)} = \cos \left( pos / 10000^{2i/d_{\text{model}}} \right)$$

其中  $pos, i$  分别表示 position 和位置编码的维度。作者选用这样的位置编码的原因：使不同位置的元素能够通过相对位置更好地与当前元素建立关系，因为对于任意固定的  $k$ ， $PE_{pos+k}$  可以表示为  $PE_{pos}$  的线性函数。

## Why Self-Attention

| Layer Type                  | Complexity per Layer     | Sequential Operations | Maximum Path Length |
|-----------------------------|--------------------------|-----------------------|---------------------|
| Self-Attention              | $O(n^2 \cdot d)$         | $O(1)$                | $O(1)$              |
| Recurrent                   | $O(n \cdot d^2)$         | $O(n)$                | $O(n)$              |
| Convolutional               | $O(k \cdot n \cdot d^2)$ | $O(1)$                | $O(\log_k(n))$      |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$   | $O(1)$                | $O(n/r)$            |

与 RNN/CNN 相比，使用 Self-Attention 主要有三个原因：

- 计算复杂度
- 可并行的计算量
- 长范围依赖关系中的路径长度（注意：长范围的依赖不一定代表距离远）。路径越短，越容易学习到长范围的依赖。Self-Attention 中将每个位置的元素都直接与其他位置的元素相连。从 Attention 的输入和输出来看，CNN 和 Attention 之间是有一定相似性的，如 Fig.22 所示。Multi-head 能够捕捉多种关系。在较长的输入序列下，可以选择 restricted 的 self-attention，即每个位置只考虑周围固定数量的邻居

三者的比较见 Tab.19

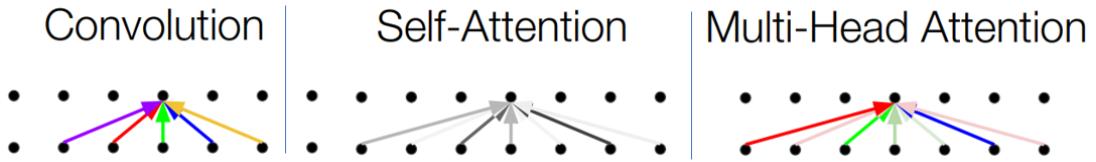


Figure 22: CNN、Attention、Multi-head Attention

### 方法解决的问题/优势

- 提出了完全基于 Attention 机制的序列建模
- 基于 Attention 的 Transformer 中的大量操作都是可以并行的，大大降低计算的时间
- 对长范围依赖关系的建模

### Attention 参考资料

- [The Annotated Transformer — Harvard Transformer tutorial](#)
- [Visualizing A Neural Machine Translation Model \(Mechanics of Seq2seq Models With Attention\)](#)
- [The Illustrated Transformer](#)

## 20 Deep Residual Learning for Image Recognition

论文地址：<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7780459>

来源：CVPR, 2016

作者：Kaiming He Xiangyu Zhang, et al.

源码：[deep-residual-networks](#)

关键词：Residual Learning, Shortcut Connection

写于：2021-03-15

该论文 [17] 主要解决的深层神经网络的训练问题。随着网络的深度的增加，模型的效果反而变差了，论文提出了 Residual Learning 的方式来训练深层的神经网络。

**问题定义** 基于深度神经网络模型在图像领域的很多任务上都取得了重大的突破，其中模型的深度起到了关键性的作用。但是随着模型深度的增加，一系列问题出现了：深度的模型只是堆叠更多的层就可以了吗？随着层数的增加，会出现梯度消失/爆炸问题，这个问题可以通过不同的归一化来解决。但还有一个问题：degradation — 模型退化。Degradation 是论文解决的问题。那么什么是模型退化呢？

考虑一个已经训练好的模型，它能达到一定的效果，为了让效果更好，我们在训练好的模型的基础上增加层，再进行训练。按照常理，新增的层就算学习到了恒等映射模型也不会退化的，但是一些实验显示，增加层后的模型的效果不但没有提升反而还下降了！模型退化了！这表明：**在求解时，使用多个非线性层来近似恒等映射可能是比较困难的**。

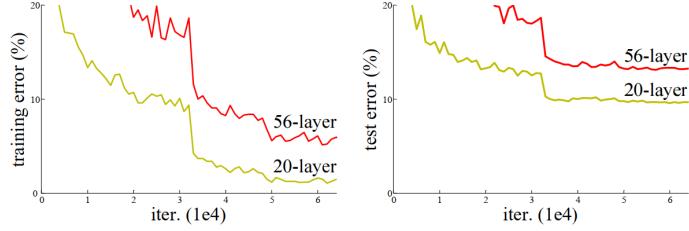


Figure 23: Degradation

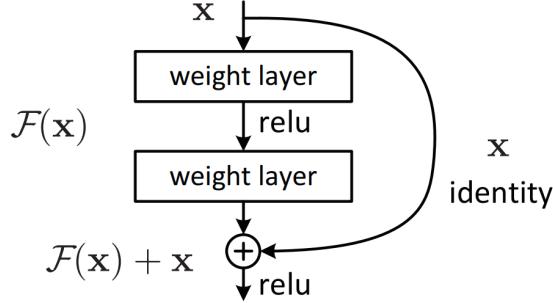


Figure 24: Residual learning: a building block

**Residual Learning** 为了解决模型退化的问题，作者提出了 *deep residual learning framework*。在加深模型时，我们通常希望新加的层能够直接学习到潜在的映射函数  $\mathcal{H}(x)$ ，residual learning 并不是如此，而是学习一个 residual mapping  $\mathcal{F}(x) = \mathcal{H}(x) - x$ 。那么原来期望学习到的映射就可以表示成： $\mathcal{H}(x) = \mathcal{F}(x) + x$ 。这里有一个假设： $\mathcal{F}$  比  $\mathcal{H}$  更容易优化。

$\mathcal{F}(x) + x$  可以用一个带有 shortcut 连接的前馈神经网络模拟，如 Fig.24 所示。如果恒等映射是最优解，那么只需要使 residual block 中层的参数为零即可。结合之前所说的非线性层难以学习 identity，residual block 不仅包含了 identity 部分，还包括了非线性部分，这样的一个映射不仅可以学习到复杂的非线性部分，还可以学习到非线性层难以学习到的线性部分（不仅是 identity，还可以是  $ax + b$ ，线性函数加非线性函数）。

Fig.24 所示的残差块可以表示为：

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

或

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s \mathbf{x}$$

以两层的残差块为例， $\mathcal{F}(x, W_i) = W_2 \sigma(W_1 x)$  表示需要学习的残差函数。当然，除了 Fig.24 所示的残差块，还可以有三层甚至更多层的残差块。论文中还对 shortcut connection 的形式进行了实验验证，第二种形式的形式效果可能会稍微好一点点，但是可以忽略，而 identity 形式的残差具有更少的参数、更低的复杂度。

## 方法的问题/优势

- 研究深度模型中的退化问题，累积的非线性层可能难以学习到线性映射
- 提出了 residual learning，助力深度模型的学习，且没有增加学习的参数
- 提出了 ResNet

## 方法的局限性/未来方向

- 累加的非线性层可能可以学习到较复杂的非线性变化，但是难以通过非线性层学习到线性变换，residual learning 弥补了这一缺点

## 21 DeepGCNs: Can GCNs Go as Deep as CNNs?

论文地址: <https://arxiv.org/pdf/1904.03751.pdf>

来源: ICCV, 2019

作者: Guohao Li, Matthias Muller, et al

源码: [deep\\_gcn\\_torch](#)

关键词: GCN, Point Cloud Segmentation

写于: 2021-03-16

该论文 [27] 聚焦于深度 GCN 的训练及在点云数据上的分割问题。常见的 GCN 的层数都比较少, 相关研究表明深层的 GCN 可能会引起如下问题:

- over-smoothing, 一个连通分量内的结点的表征会趋同
- 较高的复杂度, 不利于反向传播
- 梯度消失

该论文将 CNN 中的一些技巧引入深层 GCN 模型的构建, 并将其应用在点云数据分割上。

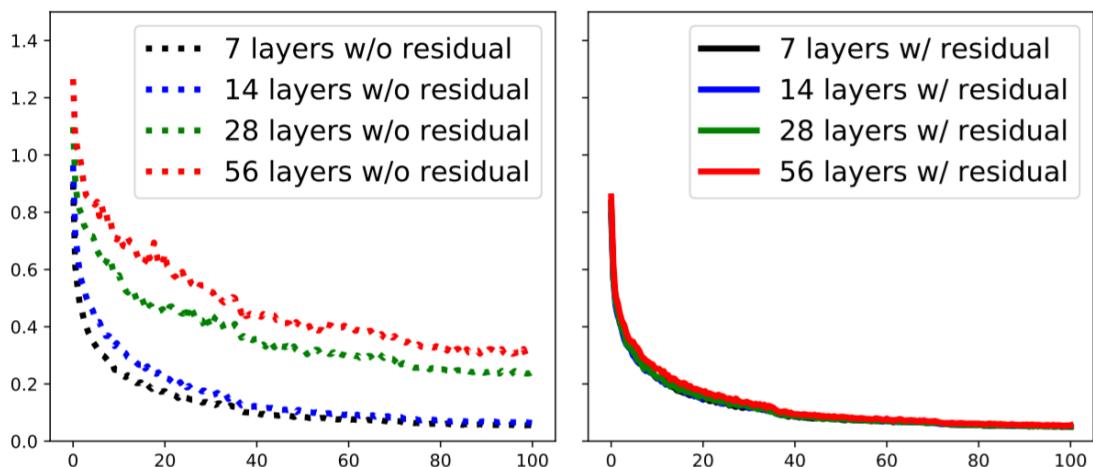


Figure 25: Training deep GCNs

**问题定义** 如 Fig.25 所示, 当 GCN 的深度增加后, 模型的损失并没有降低, 这与 [17] 中的模型退化现象很相似。怎么才可以让 GCN 像 CNN 一样, 能够搭建出很深的模型呢?

**DeepGCN** 作者从 CNN 中引入了三个技巧来解决这个问题: 1) Residual learning; 2) Dense Connection; 3) Dilated Aggregation。

**Residual Learning for GCNs** 单纯地堆积卷积层对 GCN 地效果并没有很大的提升, 其中一个原因可能是随着深度的增加, 梯度难以反向传播, 阻碍了模型的学习。Residual Learning 的引入就是为了解决这个问题。与 Residual 类似, 如果原本要学习的映射为  $\mathcal{H}$ , 残差的映射为  $\mathcal{F}$ , 则残差学习表示如下:

$$\begin{aligned}\mathcal{G}_{l+1} &= \mathcal{H}(\mathcal{G}_l, W_l) \\ &= \mathcal{F}(\mathcal{G}_l, W_l) + \mathcal{G}_l = \mathcal{G}_{l+1}^{res} + \mathcal{G}_l\end{aligned}$$

第  $l$  层的输出经过  $\mathcal{F}$  的转换再与第  $l$  层的输出逐个顶点 (vertex-wise) 相加得到第  $l+1$  层的输出。

**Dense Connections in GCNs** DenseNet 的提出是为了利用层之间的稠密连接，提高网络中信息的流动，能够对层之间的特征进行重用，形式如下：

$$\begin{aligned}\mathcal{G}_{l+1} &= \mathcal{H}(\mathcal{G}_l, W_l) \\ &= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, W_l), \mathcal{G}_l) \\ &= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, W_l), \dots, \mathcal{F}(\mathcal{G}_0, W_0), \mathcal{G}_0)\end{aligned}$$

其中  $\mathcal{T}$  是一个逐顶点的连接操作，将输入的 graph 的结点的特征与中间各个 GCN 层的输出进行拼接。

**Dilated Aggregation inn GCNs** 空洞卷积弥补了 pooling 操作对空间信息的损失，在增大感受野的同时保留了位置信息。在改论文中，作者使用 Dilated Aggregation 来为每个结点聚集信息，以论文中进行的点云分割实验为例 — 使用 Dilated k-NN 为每个顶点生成它的邻居。在每个 GCN 层后，作者使用 Dilated k-NN 来为每个顶点生成  $k$  个邻居，这  $k$  个邻居是从与这个顶点距离最近的  $k \times d$  个顶点以步长为  $d$  选择出来的，即顶点  $v$  的邻居为： $\mathcal{N}^{(d)}(v) = \{u_1, u_{1+d}, u_{1+2d}, \dots, u_{1+(k-1)d}\}$ 。如 Fig.?? 所示，上部分为 dilated 卷积，下部分为 dilated k-NN，其中选择的  $d$  分别为 1, 2, 4。在实践中，作者还增加了一点随机性，以  $\epsilon$  的概率随机从  $k \times d$  中选择  $k$  个顶点。

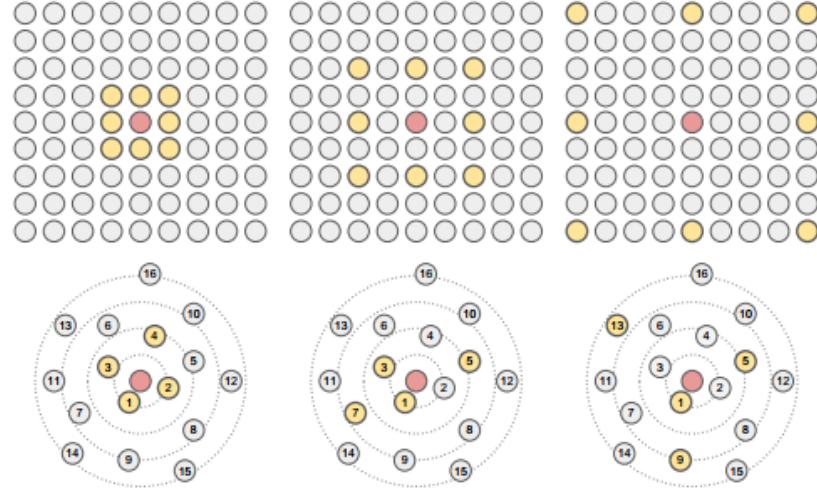


Figure 26: Dilated Convolution in GCNs

最终，论文中用于点云数据分割的深度 GCN 架构如 Fig.?? 所示。其中的区别为 GCN Backbone Block，这个 block 有三个选择：1) 普通的 GCN；2) ResGCN；3) DenseGCN。

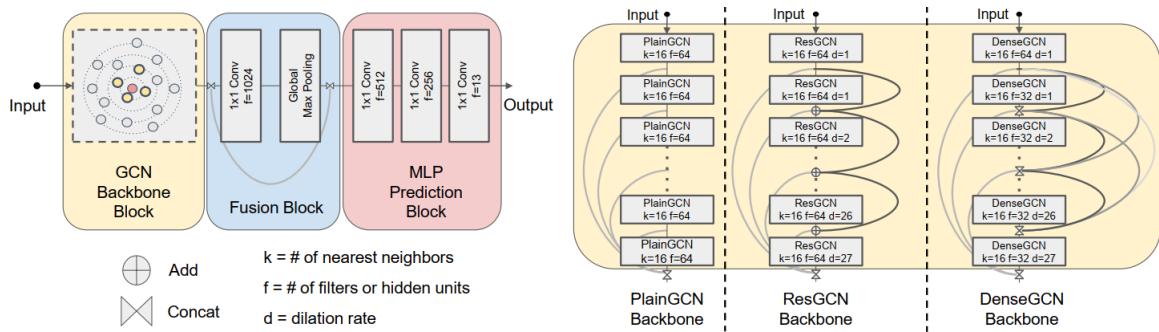


Figure 27: DeepGCN

## 总结

- 将 CNN 中的技术引入到 GCN 中，使得训练深层的 GCN 模型成为可能
- 将 GCN 应用于点云数据分割

## 22 Combining Label Propagation and Simple Models OUT-PERFORMS Graph Networks

论文地址: <https://arxiv.org/pdf/2010.13993.pdf>

来源: ICLR, 2021

作者: Qian Huangz, Horace He, et al.

源码: [CorrectAndSmooth](#)

关键词: **Label Propagation, Transductive node classification**

写于: 2021-03-25

该论文 [19] 针对 GNN 的解释性不足, GNN 模型越来越大的问题, 提出了“小米加步枪” — 用简单模型得到和 GNN 一样的水平。针对 transductive 的节点分类问题, 作者使用浅层模型加上 LP (Label Propagation) 和两个后处理方法 — Correct & Smooth, 达到或超过了 GNN 在一些 transductive 结点分类上的效果。

**问题定义** 给定一个图  $G = (V, E)$ , 结点特征矩阵为  $X \in \mathbb{R}^{n \times p}$ , 结点集  $V$  可以分为  $U, L$  分别表示无标签、有标签结点子集。标签矩阵为  $Y \in \mathbb{R}^{n \times c}$ 。目标就是基于以上所给的信息, 预测  $U$  中结点的标签。

**Correct and Smooth** 论文的方法总体过程可以分为三步:

1. 使用浅层模型, 在  $L$  上, 根据结点的特征做一个基础的预测, 即找到  $\arg \min_f l(f(x_i), y_i)$ , 得到基础预测  $Z \in \mathbb{R}^{n \times c}$ ;
2. 利用标签数据的信息去改正基础预测  $Z$ ;
3. 根据相邻结点的相似性, 对改正后的预测进行平滑;

**Simple Base Predictor** 找到一个浅层模型  $f$ , 如线性模型、MLP 等, 在训练数据 ( $L$  的子集) 上达到最小经验损失。这一步并不会使用到图的结构, 可以避免 GNN 模型的可扩展性问题。

**Correct error in base prediction with Residual Propagation** 在上一步已经有了一个基础的预测  $Z$ , 而且有标签矩阵  $Y$ , 可以根据二者在训练集 ( $L_t$ ) 上构建误差矩阵  $E$ , 也就是 residual, 即:

$$E_{L_t} = Z_{L_t} - Y_{L_t}, \quad E_{L_v} = 0, \quad E_U = 0$$

作者使用 LP 对  $E$  进行了平滑, 平滑的公式为:

$$\hat{E} = \arg \min_{W \in \mathbb{R}^{n \times c}} \text{trace}(W^T(I - S)W) + \mu \|W - E\|_F^2$$

其中  $S$  为  $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ 。上式的第一项保证  $E$  在  $G$  上的平滑, 第二项保证  $\hat{E}$  不会偏离  $E$  太多。上式可以通过迭代求解, 迭代式为:  $E^{(t+1)} = (1 - \alpha)E + \alpha SE^{(t)}$ , 其中  $\alpha = \frac{1}{1+\mu}$ ,  $E^{(0)} = E$ 。得到了  $\hat{E}$  然后呢? 当然是修正  $Z$  啦!  $Z^{(r)} = Z + \hat{E}$ 。

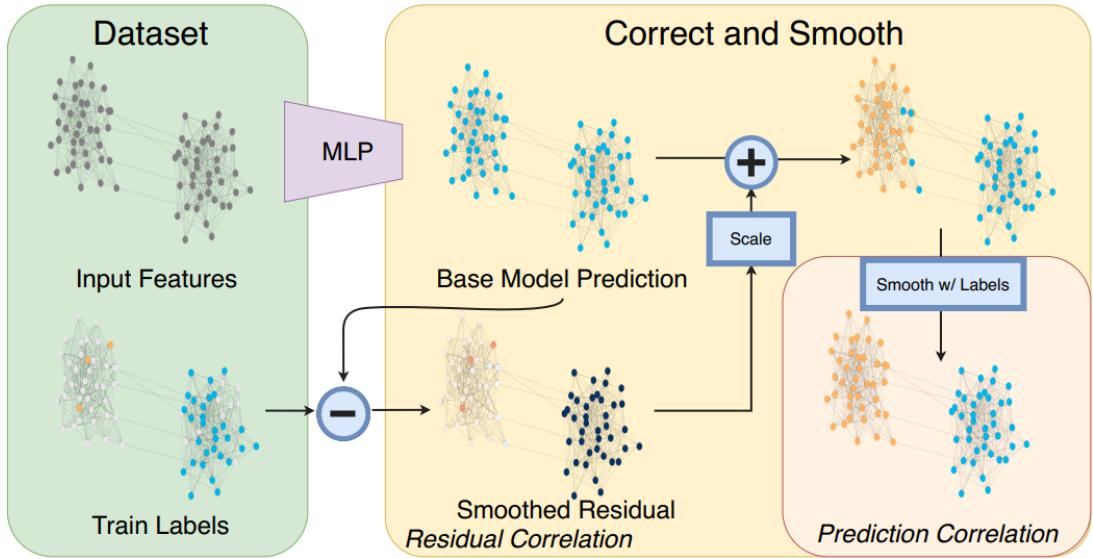


Figure 28: Correct and Smooth

那么问题来了，为什么要这样呢？*Z* 中的误差会沿着图中的边传播，结点 *i* 处的误差会使得 *i* 的邻居产生类似的误差，应当将这种误差 — 不确定性传播出去，即文中的 **error-correlation**！

但是有一个问题：

$$\|E^{(t+1)}\|_2 \leq (1 - \alpha)\|E\| + \alpha\|S\|_2 \|E^{(t)}\|_2 = (1 - \alpha)\|E\|_2 + \alpha \|E^{(t)}\|_2$$

因此  $\|E^{(t)}\|_2 \leq \|E\|_2$ ，可见在迭代求解过程中误差信息有一定损失，因此作者提出了对误差信息进行缩放。作者提出了两种缩放方法：

- Autoscale。因为在  $L_t$  上的误差是知道的，因此以  $L_t$  上的平均误差为比例进行缩放。即  $Z_{i,:}^{(r)} = Z_{i,:} + \sigma \hat{E}_{:,i} / \|\hat{E}_{:,i}^T\|_1$  for  $i \in U$ ，其中  $\sigma = \frac{1}{|L_t|} \sum_{j \in L_t} \|e_j\|_1$ ，其中  $e_j$  为  $E$  的  $j-th$  行；
- Scaled Fixed Diffusion。在迭代过程中保持标签数据上的误差不变，即  $E_U^{(t+1)} = [D^{-1}AE^{(t)}]_U$ ,  $E_L^{(t)} = E_L$ ；

**Smoothing final prediction with Prediction Correlation** 通过上一步已经得到了  $Z^{(r)}$ ，这一步是对  $Z^{(r)}$  的“平滑”。这一步的动机是：相邻的结点相似，更可能具有相同的标签，即文中的 **prediction correlation**，这一步再次借助 LP 对结点的标签分布进行平滑。

首先构造一个  $G$  中结点的标签分布  $P \in \mathbb{R}^{n \times c}$ ,  $P_{L_t} = Y_{L_t}$ ,  $G_{L_v, U} = Z_{L_v, U}^{(r)}$ 。接着就是平滑了，也就是一个迭代的过程： $P^{(t+1)} = (1 - \alpha)P + \alpha SP^{(t)}$ ，一直迭代到收敛得到最终的收敛  $\hat{Y}_{ij}$ 。

## 总结

- 没用一股脑地上深度模型（当然，GNN 大多不是深层的）。浅层模型 +LP 在 transductive node classification 上的效果直追/超过了一些 GNN 模型；
- 论文中的方法效果不错，且大大降低了模型的参数量、学习时间，速度快；
- 不足的是只能进行 transductive 的任务，目前的趋势是 inductive learning；
- 不知道该论文的方法在其他任务上会有怎样的效果；

## 23 On the Bottleneck of Graph Neural Networks And Practical Implication

论文地址: <https://openreview.net/pdf?id=i80OPhOCVH2>

来源: ICLR, 2021

作者: Uri Alon, Eran Yahav

源码: [bottleneck](#)

关键词: GNN, Over-Squashing

写于: 2021-03-28

该论文 [1] 针对 GNN 中的远距离信息的传播问题提出了一个新的解释。文中讨论的问题是: GNN 在聚集/利用远距离结点的信息时存在困难/一个瓶颈。论文针对这个问题进行了讨论和分析，并通过一些例子验证了这个问题。

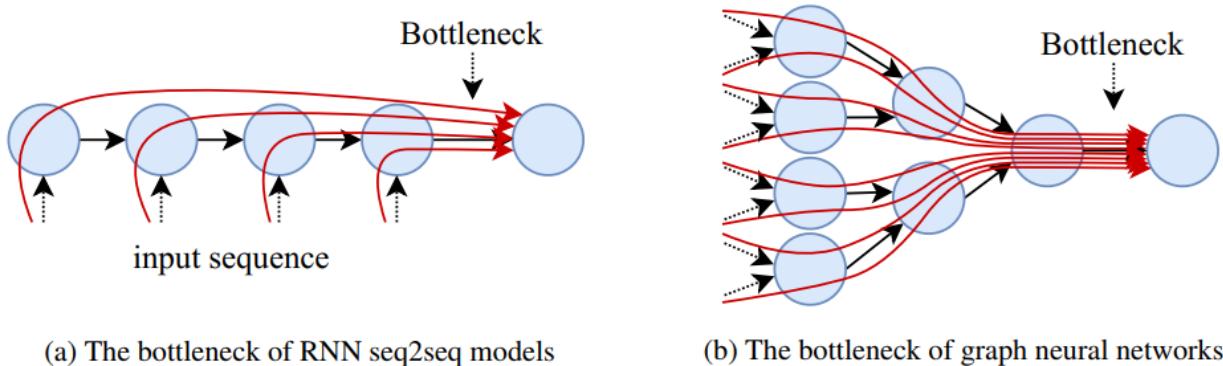


Figure 29: The bottlenecks of Seq2seq and GNN models

**Over Squashing** 作者将这个 Bottleneck 称为 Over Squashing, 如 Fig.29 所示。什么是 Over Squashing 呢? 这就要从 GNN 的工作方式开始说起了。

GNN 的工作方式是不断的将邻居结点的信息汇集 (aggregate) 起来, 再进行后续的操作, 通常一层汇聚的就是 1-hop 的邻居的信息,  $k$  层就可以汇聚到  $k$ -hop 邻居的信息。随着层数的增加, 一个结点的  $k$ -hop 邻居的数量将会呈指数的形式增长, 但是与前几层相比, 更多的信息被压缩到了一个定长的向量中, 这就是 over squashing。

在一些 graph 的任务中, 不仅需要依靠局部的信息, 如 1-hop 或者比较近的邻居的信息, 同时还需要远距离的结点的信息。作者对任务所需要的邻居结点的距离定义为 *problem radius*。目前大部分使用的 GNN 都是比较浅层的, 比如 GCN 的开山之作中就只有两层。作者在实验中还发现 GCN、GIN 比 GAT 等一些 GNN 模型更容易出现这个问题, 显然, 注意力机制在这里起了作用。

over squashing 是指随着层数增加, 指数速度增加的邻居的信息被过度压缩进了一个定长向量中, 还有一个问题就是, 对于最短路径大于 GNN 层数的情况, 这个时候远距离的结点信息就不能传过来了, 作者将其定义为 under reaching。

作者在实验中发现, 当 *problem radius* 越大, 则结点表征就需要更大的维度。

### 总结

- 将详细地研究了 GNN 中的远距离结点的信息的传播问题

- 之前的一些工作与这个有关，如 [27]
- 有哪些较好地方法可以解决 over squashing 问题呢？
- 对于远距离结点地信息传播问题，能否通过在汇聚邻居信息时，随机采样较远的结点，作为远距离结点信息地补充
- 有哪些任务是需要远距离结点信息的呢？
- 能否用 Transformer 来作为 GCN 中的 aggregate 操作？

## 24 Representation Learning via Invariant Causal Mechanism

论文地址： <https://arxiv.org/pdf/2010.07922.pdf>

来源： ICLR, 2021

作者： Jovana Mitrovic, Brian McWilliams, et al.

关键词： Self-Supervised learning, Contrastive Learning, Causal

写于： 2021-03-31

该论文 [40] 从因果的视角下解释自监督学习，将数据生成过程看成由两个因子控制的过程。并且提出了 RELIC (Representation Learning with Invariant Mechanisms)，能够学到尽量与下游任务无关的表征。

**问题定义**  $X$  表示未标记的数据， $\mathcal{Y} = \{Y_t\}_{t=1}^T$  表示一系列未知的下游任务。目标就是使用  $X$  学习到有利于  $\mathcal{Y}$  的表征。

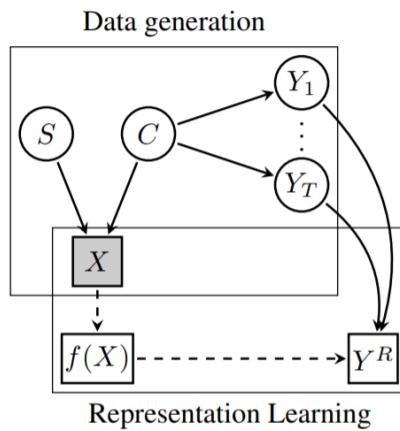


Figure 30: Data Generation in Causal interpretation

**RELIC** 该论文主要针对的一个问题是：通过自监督学习到的表征尽量与下游任务无关。为了解决这个问题，作者将数据生成看作是有因果关系的，如 Fig.30所示。其中  $S$ ,  $C$  分别表示 Style 和 Content。并有三个前提假设：1) 数据是由  $S$ ,  $C$  生成的（有点变分推断 [25] 的味道）；2) 只有  $C$  与下游任务是相关的；3)  $C$  和  $S$  是不相关的。

在这里， $S$  表示不同的数据增强方式。为了使数据的表则在不同的 Style 下是与下游任务是无关的，需要满足如下形式：

$$p^{do(a_i)}(Y^R | f(X)) = p^{do(a_j)}(Y^R | f(X)) \quad \forall a_i, a_j \in \mathcal{A}$$

其中  $p^{do(a_i)}$  表示施加 Style  $a_i$  后的数据分布,  $\mathcal{A}$  表示数据增强方式集合, 即所有的 styles。为了在不同的数据增强方式下对下游任务都是无关的, 作者还提出了一个正则化项, 最终的目标函数如下:

$$\mathbb{E}_{X \sim a_{lk}, a_{qt} \sim \mathcal{A} \times \mathcal{A}} \sum_{b \in \{a_{il}, a_{qt}\}} \mathcal{L}_b(Y^R, f(X)) \quad s.t. \quad KL(p^{do(a_{lk})}(Y^R | f(X)), p^{do(a_{qt})}(Y^R | f(X))) \leq \rho$$

## 总结

- 用因果解释自监督学习的过程, 并且提出了使表征与下游任务无关的学习目标
- 看的不是很懂, 还是太菜了 joy

## 25 U-Net: Convolutional Networks for Biomedical Image Segmentation

论文地址: <https://arxiv.org/pdf/1505.04597.pdf>

来源: LNCS, 2015

作者: Ronneberger, Olaf and Fischer, et al.

关键词: **Image Segmentation, Biomedical Image**

写于: 2021-04-19

该论文 [46] 提出的 U-Net, 是较早将卷积神经网络应用到图像语义分割的模型。U-Net 是在 FCN[31] 的基础上构建的。U-Net 包含 contracting path 来提取图像特征/上下文、expanding path 来进行精确的分割。

**Motivation** CNN 通常用被用于分类, 并且需要大量的数据。然而在医学图像处理任务中, 不仅是对图像分类, 还需要对图像进行逐像素的分割, 而且通常无法获得大量的医学图像数据。在之前的一些方法中, 将像素周围的局部区域划分为一个 patch 来预测一个像素的类别, 这样获得的 patch 的数量远远大于图像的数量。但是这种方法有两个明显的缺点: 1) 速度慢, 因为需要单独计算每个 patch 来预测像素类别, 而且 patch 之间存在巨大的冗余; 2) 预测的准确率受所使用的 patch 的 size 的影响。还有一些方法使用不同层级的 feature map 来进行分割 (如金字塔模型)。

鉴于已有方法的缺陷, U-Net 在 FCN[31] 的基础上进行修改和扩充, 使得 U-Net 能够在数据量小的情况下得到更好的、高精度的分割效果。

**U-Net** U-Net 结构如 Fig.31 所示。U-Net 整体可以分为两部分, 左边的压缩路径部分和右边的扩展路径部分。

在压缩路径上, 输入的源图像经过卷积、max-pooling, 尺寸逐渐变小, 尺寸每为原来的二分之一, 通道数则变为原来的两倍。在扩展路径上, feature map 经过 up-conv、卷积, 尺寸逐渐增大, 尺寸每变为原来的两倍, 通道数变为原来的二分之一。除了左右两边的路径, U-Net 还会将左边的压缩路径上的 feature map 裁剪一部分与右边的扩张路径同层级的 feature map 在通道上进行拼接, 如 Fig.31 中间灰色箭头所示。

到目前为止, U-Net 都是比较简洁明了的, 现在来更深入地挖掘一下。

**Overlap-tile strategy** 使用 U-Net 对源图像进行分割时, 我们可以看到输出的分割结果比源图像要小, 为了使得 U-Net 能够对任意大小的图像进行无缝的 (seamless) 分割, U-Net 使用了 Overlap-tile strategy, 如 Fig.32 所示。对于很大的图像, 进行分割时可能需要将其进行划分成符合网络输入的块, 原本在源图像内处于内部的区域可能成为了块的边界, 在对块进行分割时, 可能会产生缝隙/无法与相邻块的分割结果对齐。为了解决这个问题, 也为了维持分割结果的 size, U-Net 使用了 Overlap-tile strategy, 以块的边界为轴进行镜像对称来对块进行 padding (这一点不是很确定 confused, 论文中原文是: *only use the valid part of each convolution*)。

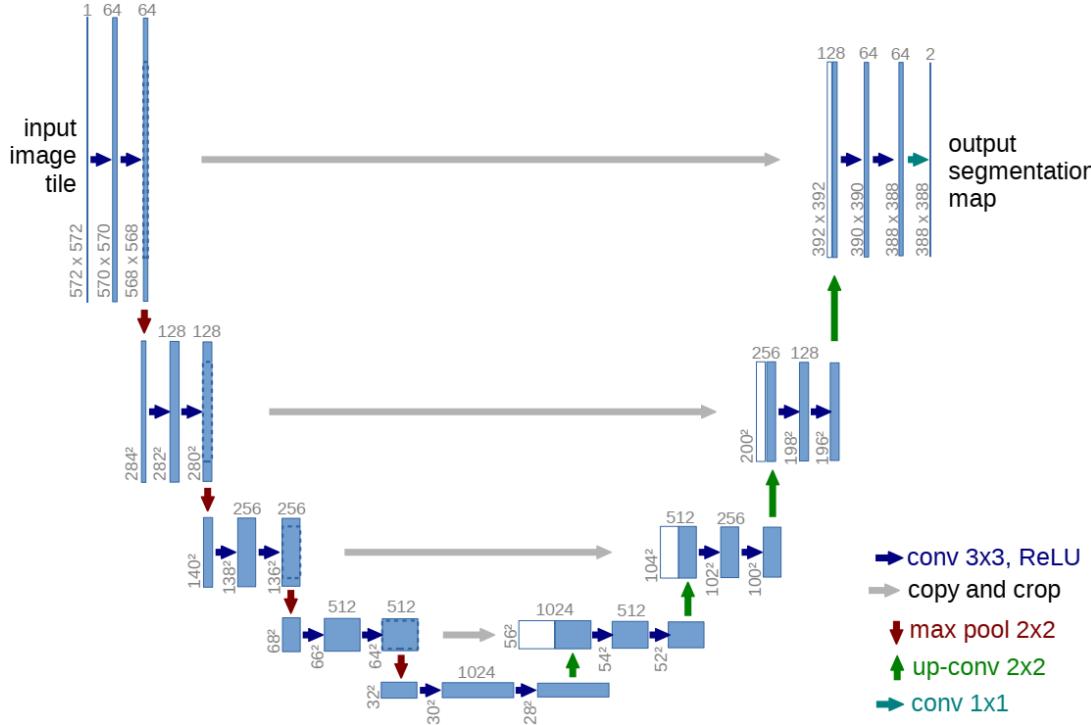


Figure 31: U-Net architecture

**损失函数** 为了让 U-Net 对边界有更好的分割效果、对与相邻的同类区域也有较好的分割效果，U-Net 使用了带权重的交叉熵损失函数，该损失函数在每个像素处对  $p_{l(x)}(x)$  与 1 的偏差进行了惩罚：

$$E = \sum_{x \in \omega} w(x) \log(p_{l(x)}(x))$$

其中  $x \in \Omega, \Omega \subset \mathbb{Z}^2$ ,  $x$  表示像素位置,  $l : \Omega \rightarrow 1, \dots, K$  表示每个像素的真实标签。 $p_{l(x)}$  是 softmax 函数,  $w : \Omega \rightarrow \mathbb{R}$  表示像素的权重。使用  $w$  有两个原因：1) 类别不平衡；2) 强调分割边界。 $w$  定义如下：

$$w(x) = w_c(x) + w_0 \cdot \exp\left(-\frac{(d_1(x) + d_2(x))^2}{2\sigma^2}\right)$$

其中  $w_c : \Omega \rightarrow \mathbb{R}$  用于解决类别不平衡的问题,  $d_1 : \Omega \rightarrow \mathbb{R}$  表示  $x$  到最近的细胞的距离（因为该论文是针对细胞分割任务的，在其他任务中，个人认为可以理解为到最近的分割目标距离）， $d_2 : \Omega \rightarrow \mathbb{R}$  表示  $x$  到第二近的细胞的距离。 $w_0, \sigma$  为超参数。

**Data Augmentation** 由于训练数据较少，作者对数据进行了平移、旋转等增强，最重要的是弹性变形增强。

## 总结

- U-Net 是较早使用多尺度特征进行图像分割的算法
- U 形结构对后续的图像分割模型有很大的启发
- 由于 valid convolution，输入与输出的大小不一致

## 26 UNSUPERVISED REPRESENTATION LEARNING FOR TIME SERIES WITH TEMPORAL NEIGHBORHOOD CODING

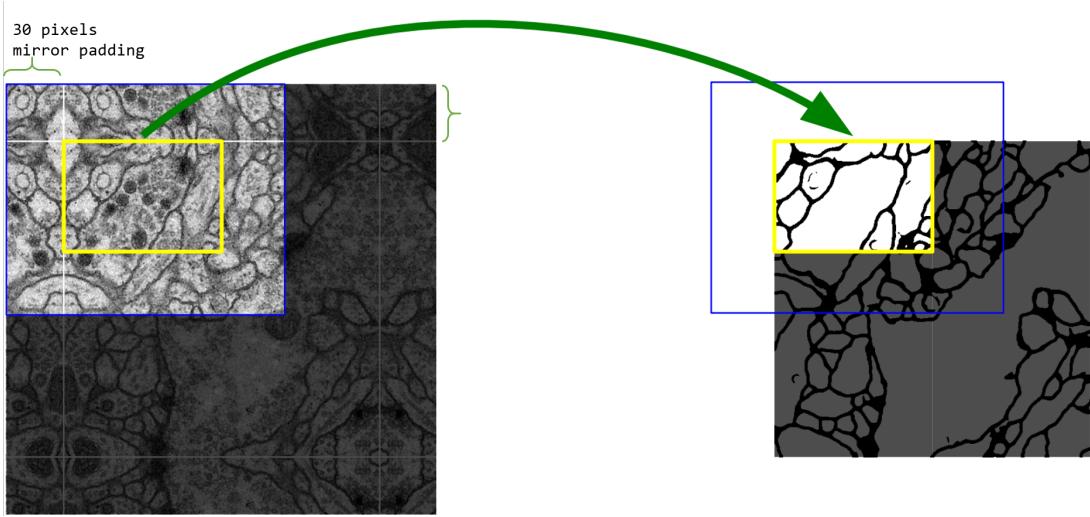


Figure 32: Overlap-tile strategy

论文地址: <https://arxiv.org/pdf/2106.00750.pdf>

来源: ICLR, 2021

作者: Sana Tonekaboni, Danny Eytan, Anna Goldenberg

源码: [TNC](#)

关键词: time series, unsupervised representation learning

写于: 2021-06-16

该论文 [54] 针对的是 (非平稳) 时间序列数据的无监督表征问题, 论文的 Motivation 来自 Healthcare 领域, 患者的状态会在不同的临床状态之间转移。文中采用了对比学习的思想。

**问题定义** 给定一个时间序列样本  $X \in R^{D \times T}$ , 其中  $T$  该样本所跨越的时间长度,  $D$  表示样本在每一个时间步的特征维度。 $X_{[t-\frac{\delta}{2}, t+\frac{\delta}{2}]}$  表示  $X$  上的长度为  $\delta$  的时间窗口, 因为该窗口以  $t$  为中心, 我们表示为  $W_t$ 。论文要解决的问题就是通过无监督学习得到  $W_t$  的表征。

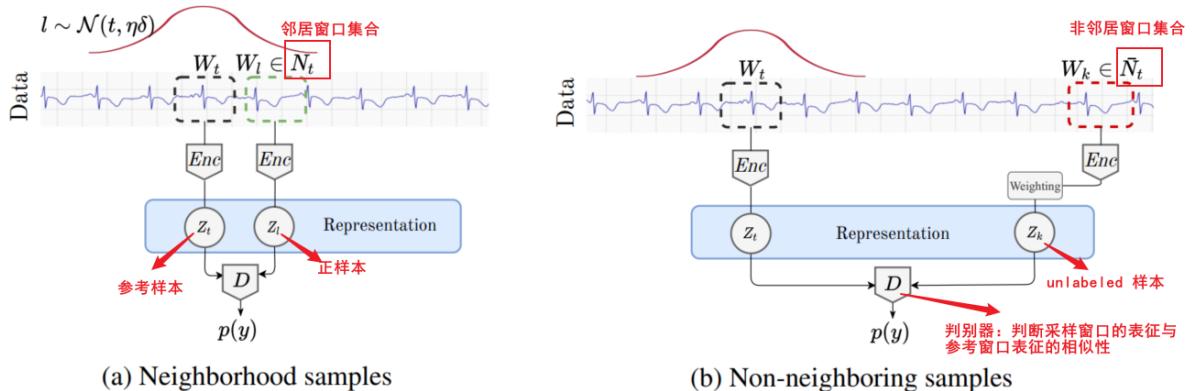


Figure 33: TNC Architecture

**TNC 思路** TNC 框架如 Fig.33 所示。在问题定义中已经看到, 论文是要学习一个时间窗口的表征, 加上论文中使用了对比学习, 因此很容易想到: 对于一个  $W_t$ , 需要采样正样本  $W_l$  与负样本  $W_k$ 。既然如此, 怎么采样就成了一个关键问题。

文中定义了  $W_t$  的 Temporal Neighborhood ( $N_t$ )，以及 Non Temporal Neighborhood ( $\bar{N}_t$ )。显而易见， $W_l \in N_t$ ,  $W_k \in \bar{N}_t$ 。既然如此，那么怎么定义 Temporal Neighborhood 呢？

说明白了， $N_t, \bar{N}_t$  都是窗口的集合，那么只要给定窗口长度以及窗口中心就确定了一个窗口，通常窗口长度都是固定的，那么只需要确定窗口中心即可。论文中对  $N_t$  中的窗口中心  $t^*$  定义为： $t^* \sim \mathcal{N}(t, \eta \cdot \delta)$ ，其中  $\eta$  表示邻居的范围（有点不解）， $\delta$  表示窗口长度。作为时间序列的信号，通常有一个假设：信号局部是平稳的，即是光滑的。因此  $N_t$  中的窗口与  $W_t$  相似。 $\eta$  对于每个  $t$  都是不同的， $\eta$  要尽量反映出信号保持静止（大致静止）的时间跨度，因此论文中使用了 Augmented Dickey-Fuller (ADF) 统计检验来为每个  $W_t$  计算  $\eta$ 。

$\bar{N}_t$  的采样，则是从与  $t$  有一定距离（从代码中可以看出这个距离是和  $\eta$  有关的）的范围内采样得到的。论文中并没有直接将  $\bar{N}_t$  中的样本当作负样本。原因是 sample bias：从数据分布中随机采样的负样本有可能与参考样本是相似的。论文中将  $W_k \in \bar{N}_t$  视作 unlabeled 样本。从实践的角度来看：用一个参数  $w$  表示 unlabeled 样本属于正样本的概率，因此一个 unlabeled 样本表示  $w$  个正样本与  $1-w$  个负样本的结合，在计算损失的时候，unlabeled 样本会分别作为正负样本与参考样本计算损失，再分别乘以  $w, 1-w$  作为这个 unlabeled 样本的损失。

$$\mathcal{L} = -\mathbb{E}_{W_t \sim X} \left[ \mathbb{E}_{W_l \sim N_t} [\log \underbrace{\mathcal{D}(Z_t, Z_l)}_{\mathcal{D}(\text{Enc}(W_t), \text{Enc}(W_l))}] + \mathbb{E}_{W_k \sim \bar{N}_t} [(1-w_t) \times \log \underbrace{(1 - \mathcal{D}(Z_t, Z_k))}_{\mathcal{D}(\text{Enc}(W_t), \text{Enc}(W_k))} + w_t \times \log \mathcal{D}(Z_t, Z_k)] \right]$$

**流程：**获取一个参考样本  $W_t$ ，在获取一个正样本  $W_l$  以及一个 unlabeled 样本  $W_k$ ，分别传入一个编码器（通常是序列模型，可以是 RNN，因为每个样本就是一个时间序列样本）中，分别得到表征  $Z_t, Z_l, Z_k$ 。用一个判别器（其实就是一个分类器）分别计算  $(Z_t, Z_l), (Z_t, Z_k)$  一个分数（该样本对相似的度量），再计算对比损失，特别要注意  $(Z_t, Z_l)$  的损失哦！

## 总结

- 文中对 unlabeled 的想法很有趣
- 文中的实验通过聚类以及 trajectory 来发现序列随时间的变化，这也是我之前考虑过的一个问题
- 不知道用在图数据集上怎么样，挖掘图数据的变化，可以是 graph-level 的或者是 node-level 的

## 27 Graph Similarity

### 27.1 graph2vec: Learning Distributed Representation of Graphs

论文地址：<https://arxiv.org/abs/1707.05005>

源码地址：[graph2vec](#)

关键词：Graph Kernels, doc2vec, representation learning

写于：2020-09-30

论文 [41] 提出了图的 embedding 方法—graph2vec，该方法借鉴 word2vec, doc2vec 的思想，以 rooted subgraph 作为图的元素，通过最大化似然概率来得到每个图的定长的 embedding 向量。

如果熟悉 word2vec 的话，其实很容易就理解这篇论文的方法了（虽然 graph2vec 是在模仿 doc2vec，但 doc2vec 开起来像在模仿 word2vec）。因为 graph2vec 是在 word2vec 基础上做的，它们之间存在一个比较直白的对应关系，厘清它们之间的对应关系就明白 graph2vec 了。

在介绍 doc2vec 和 graph2vec 的对应关系之前，先介绍一下 rooted graph 的概念。rooted subgraph 是一个类似于树状的结构，可以看作一棵多叉树，树中的结点就是图中的结点，结点按照图中的邻接关系相连。可以根据图中的每个结点  $v$  生成一棵 rooted subgraph，以  $v$  为根，它的子结点为它的邻居结点，每个子结点又以自己

的邻居结点为子结点，如此递归定义下去就是以  $v$  为根的 rooted subgraph。接下来介绍 graph2vec 和 doc2vec 之间的概念对应关系。

在 doc2vec 中，每个 doc(文档) 被看做词序列的集合，优化文档向量的基础就是：最大化词序列和文档的共现率。graph2vec 将一个 graph 看做一个文档，rooted graph 看做文档中的词序列。所以优化 graph embedding 的基础就是：最大化 rooted subgraph 和 graph 的共现率。与 word2vec 类似，graph2vec 中也使用了负采样的方法对算法进行优化。

与最近使用 GNN 来进行 graph embedding 的方法相比，graph2vec 是 transductive 的，对于未见过的 graph 需要重新运行一遍算法来生成 embedding，与 Deepwalk 很像。我这里有一个想法，rooted subgraph 能否通过其他形式来替代，比如 node2vec 中的游走策略生成的子图。

## 27.2 Convolutional Set Matching for Graph Similarity

论文地址：<https://arxiv.org/abs/1810.10866>

slides：[Convolutional Set Matching for Graph Similarity](#)

关键词：**Graph Similarity, GCN, CNN**

写于：2020-10-10

该论文 [5] 主要解决的问题是：给定两个图  $\mathcal{G}_1, \mathcal{G}_2$ ，计算它们的相似性。图相似性计算是图数据搜索中的难点和核心点，由于图的同构性，衡量两个图的相似性是 NP-hard 问题。论文中提出了 GSimCNN (Graph Similarity Computation via Convolutional Neural Network) 来解决这个问题。

**GSimCNN 思路** 该方法使用了图像识别中 CNN 的方法，通过 GCN 可以得到每个 graph 的多个结点特征矩阵 (GCN 有几层就有几个节点特征矩阵)，这样就得到了  $\mathcal{G}_1, \mathcal{G}_2$  的节点特征矩阵，再根据这两个结点特征矩阵构造一个相似矩阵 (Similarity Matrix)，构造方法是从  $\mathcal{G}_1, \mathcal{G}_2$  各取一个结点的特征矩阵做内积得到相似矩阵中的一个元素。如果 GCN 有  $k$  层，则可以得到  $k$  个相似矩阵。在经过 CNN 的一顿操作后将 CNN 的输出输入到一个全连接的 MLP 中，得到两个 graph 的相似度分数。损失函数是均方误差损失函数（对于  $\mathcal{G}_1, \mathcal{G}_2$ ，会有一个实际的相似度分数）。可参考 Fig.34。

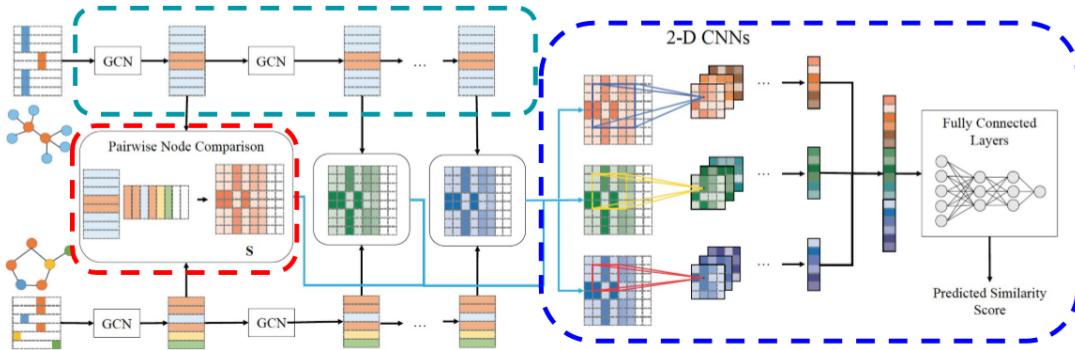


Figure 34: GSimCNN 的三个阶段，分别用不同颜色的虚线圈出

如 Fig.34 所示，是使用 GSimCNN 计算两个图相似性的过程，不同颜色代表算法的不同阶段。

**Stage 1** GCN 阶段，使用一个多层次的 GCN 为  $\mathcal{G}_1, \mathcal{G}_2$  中的每个结点生成表征，这样在 GCN 的每一层，都能够生成两个结点特征矩阵。若 GCN 有  $k$  层，则会有  $k$  对结点特征矩阵对。**为什么要在每一层计算一个相似矩阵呢？** 因为 GCN 是基于汇集邻居的信息来表征的，第  $k$  层相当于汇集了  $k$ -hop 的邻居的信息，而图的结构相似往往是局部的，而且是在不同尺度（图的规模）上的。而不同层的结点表征相当于代表了不同尺度的结构，也就能在不同尺度上进行图的相似性比较。

**Stage 2** 使用每一层的结点特征矩阵对，通过内积运算得到一个相似矩阵。因为 GCN 有  $k$  层，则会得到  $k$  个相似矩阵。

**Stage 3** 基于**Stage 2**得到的  $k$  个相似矩阵，使用多个卷积核（Fig.34中使用了 3 个 2-D 卷积核）来对多个相似矩阵进行卷积，最后将多个卷积核的结果向量进行拼接，输入到全连接的 MLP 中计算相似度分数。

在实验方面，作者使用了两类的 baseline，一类是基于组合优化的近似 GED (Graph Edit Distance，相关论文表明当图地结点超过 16 时，GED 将变得难以计算 [2]，因此也产生了很多近似算法) 的算法，另一类的基于神经网络的算法。在这些算法中，GED 的算法作为基准，其 mse 是 0，但是复杂度太高，时间成本太高，GSimCNN 与其他算法相比在 mse 和时间上都取得较好的效果。

论文中还有一些其他细节的处理，比如：对于不同大小的图的特征矩阵会进行 padding，针对结点的排序使用了 [68] 的 BFS 的思路等。

至于论文的题目中的 **set matching**，因为论文中使用的是 Set matching 的思路来计算两个图的相似性，一个图的所有结点的表征就是一个 set。那为什么不适用 graph-level 的表征来直接计算图的相似性呢？（在论文的实验比较中有几个方法使用的就是 graph-level 的表征，如：EmbAvg, GCNMean, GCNMax，这几个方法在通过结点表征得到图表征时的方法不一样，如算术平均结点的表征作为图表征等）论文中给出的原因如下：

- 没有使用更细粒度的结点表征
- 忽略了图级的交互（原文是 **graph level interaction**，但不确定是什么意思）

也正是因为以上原因，论文将图相似性以 set matching 的角度来解决。个人认为论文是在更细粒度对两幅图的相似性进行了比较（**能否进行更细粒度的比比较呢？** 比如利用结点的属）。

论文中对 GSimCNN 得到的相似矩阵进行了可视化，发现相似或很不相似的图的相似矩阵都有一些明显的特点，如 Fig.35 所示：

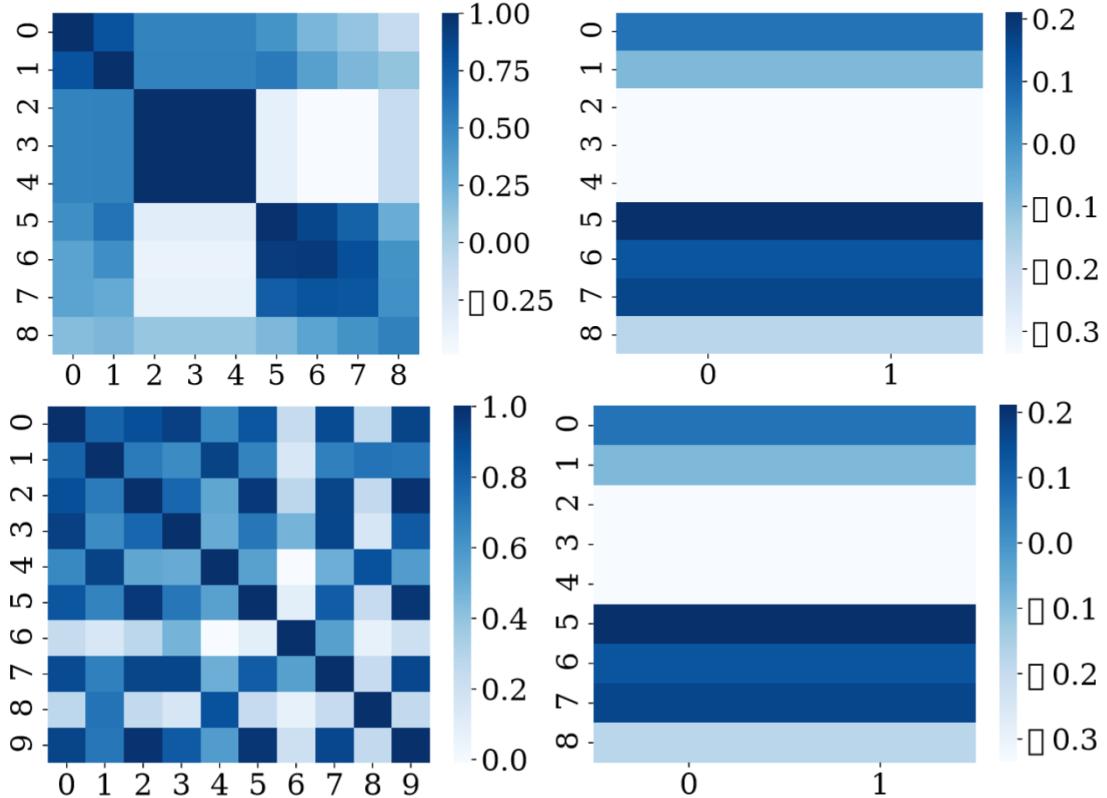


Figure 35: 左边的为相似的图的相似矩阵，右边为不相似图的相似矩阵

未来工作的方向：通过生成图的编辑序列来更好地解释计算得到的相似度（**这个可以有！**）；使用其他的结点表征网络，如 GraphSAGE；使用其他的图相似度度量方法（**比如？**）；在其他领域地数据上的泛化。

本篇论文的作者 Yunsheng Bai 在 Garph Matching 和 Graph Similarity 方面做了较多的工作，可参考[他的主页](#)。

### 27.3 SimGNN: A Neural Network Approach to Fast Graph Similarity Computation

论文地址：<https://arxiv.org/abs/1808.05689v4>

源码：[SimGNN](#)

关键词：**Graph Similarity, GCN, Graph Edit Distance**

写于：2020-10-13

该论文 [3] 与27.2的作者相同，也是为了解决图相似性计算的问题，该论文的方法/模型与27.2中的方法/模型类似。论文中提出了 SimGNN (Graph Similarity Computation via Graph Neural Network) 来解决这个问题。

**SimGNN 思路** 该方法针对图相似性的计算提出了两个策略 (strategy)，能够分别计算  $\mathcal{G}_1, \mathcal{G}_2$  的相似性，论文中将这两策略结合起来一起使用。Strategy 1 是通过 GCN 得到 graph-level 的图表征，基于图表征计算相似度。Strategy 2 是利用图中的结点表征进行相似度的计算。Strategy 1/2 的过程可参考 Fig.36，细节如下。

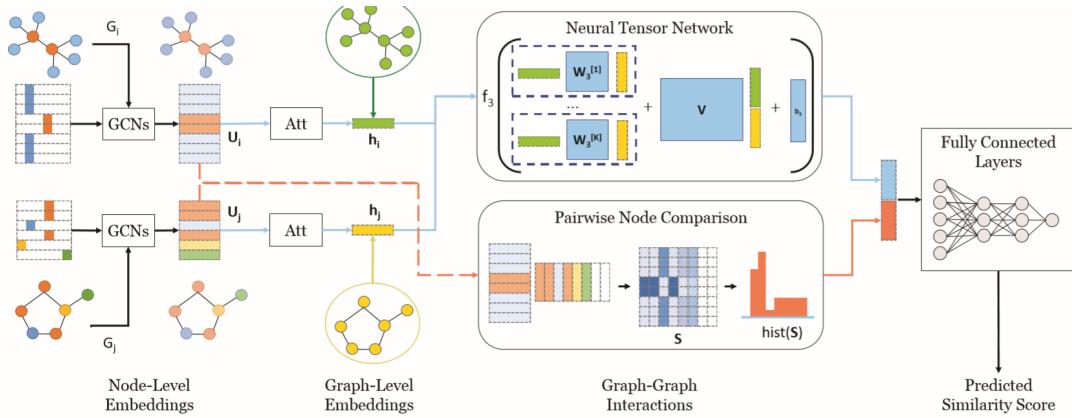


Figure 36: SimGNN

**Strategy 1** Strategy 1 是通过 GCN 计算每个结点的表征，再通过注意力的方法计算得到图的表征，该过程又可以分为 4 个阶段。

**Stage 1** 计算节点表征。如 Fig.36所示，通过 GCN (论文中使用的[GCN](#)) 计算得到每个结点的表征。

**Stage 2** 计算图表征。SimGNN 是以结点表征为基础，基于注意力计算图表征的。在计算图表征之前，先计算图的上下文 (global graph context) — 上下文向量  $\mathbf{c} = \tanh(\frac{1}{N} \mathbf{W}_2 \sum_{n=1}^N \mathbf{u}_n)$ ，其中  $\mathbf{W}_2$  是待学习得参数。对于任意一个结点  $\mathbf{u}_n$ ，它的注意力 (权重) 计算方式是： $attention(\mathbf{u}_n) = \sigma(\mathbf{u}_n^T \mathbf{c})$ 。最终图  $\mathcal{G}$  的表征  $\mathbf{h} = \sum_{n=1}^N attention(\mathbf{u}_n) \mathbf{u}_n$ 。

论文中特别提到了一个点 — 没有对  $attention(\mathbf{u}_n)$  进行归一化，即虽然  $attention(\mathbf{u}_n), \forall n \in \{1, \dots, N\}$  是在 0 到 1 之间的，但是没有经过一个类似 softmax 的归一化。这是为什么呢？难道总的权重不应该为 1 吗？论文中给出的解释是：让最后得到的图表征反映图的大小，这对相似性计算是很重要的。很巧妙，**并不是所有的情况都需要归一化！**

**Stage 3** 使用图表征计算 Graph-Graph interaction。这里的图交互（个人翻译），指的是通过图表征计算一个相似度向量。具体操作：使用 Neural Tensor Network(NTN) 来对图表征之间的关系进行建模。则两个图表征之间的关系表示为  $g(\mathbf{h}_i, \mathbf{h}_j) = f(\mathbf{h}_i^T \mathbf{W}_3^{[1:K]} \mathbf{h}_j + \mathbf{V} \begin{bmatrix} \mathbf{h}_i \\ \mathbf{h}_j \end{bmatrix} + \mathbf{b}_3)$ 。其中  $\mathbf{W}_3^{[1:K]}$  是一个张量，包括  $\mathbf{V}$   $\mathbf{b}_3$  都是待学习的参数， $K$  是超参数，最后  $g(\mathbf{h}_i, \mathbf{h}_j)$  的结果就是一个  $\mathbb{R}^K$  中的相似度向量。

**Stage 4** 计算图相似度。从 stage 3 得到两幅图的相似度向量后使用全连接的 MLP 计算相似度。

以上 4 个阶段就是 Strategy 1 的全过程，使用图标表征的 interacting 能够反映相似度基于这样一个假设：好的图表征应该能够编码图的结构和特征信息，可以通过图表征的 interacting 来预测图之间的相似性（**为什么通过 interacting 就能反映图之间的相似性呢？**）。

**Strategy 2** 使用 GCN 输出的结点表征，计算一个与 27.2 中类似的相似矩阵，可以得到一个更细粒度（fine-grained）的相似评估。因为图的结点没有一个特定的排列，Strategy 2 使用了一种结点排列无关的方法 — 提取相似矩阵的直方图特征，来利用相似矩阵计算相似度。直方图的 bin 的数量是超参数。

将 Strategy 1 和 Strategy 2 结合起来使用时将 Strategy 1 中 stage 3 得到的相似度向量与 Strategy 2 得到的直方图特征（也是一个向量）拼接后输入到全连接的 MLP 中计算得到一个相似度。

至此，SimGNN 的过程就说完了。总的来说，Strategy 1 是从粗粒度上来计算图之间的相似性，而 Strategy 2 则是从更细粒度上计算结点相似性，Strategy 2 能够对 Strategy 1 进行补充、增强（但论文中的实验说明，使用 Strategy 2 后并没有较大的提升）。

## 方法解决的问题/优势

- SimGNN 计算相似性是结点排列无关的。也就是说，同一个图，不同的结点排列，不会影响它和其他图的相似性的计算。这主要是因为使用了相似矩阵的直方图特征
- 模型是 inductive 的，能够泛化到未见过的图。SimGNN 中使用的 GCN 是 inductive 的
- SimGNN 是 end-to-end 的模型，参数都是可学习的

## 方法的局限性/未来方向

- 在 Strategy 2 中使用直方图特征虽然是结点排列无关的，但个人认为不能充分地利用相似矩阵，这也可能是为什么 Strategy 2 效果不明显的原因。或许有更好的结点排列无关的方法来利用结点表征（并不一定要使用相似矩阵）
- 图上下文向量  $c$  的原因，为什么使用它？
- SimGNN 只使用了结点的特征，并没有**使用边的特征**，后期可以将边的特征纳入模型考虑的范畴。在一些领域，如化学领域，边的类型是很重要的
- 目前的图的相似性计算主要针对较小的图，如何计算大规模的图的相似性呢
- **如何计算更复杂的图的相似性呢，如动态图**

作者其他工作可参考[他的主页](#)。

## 27.4 Unsupervised Inductive Graph-level Representation Learning via Grpah-Graph proximity

论文地址: <https://arxiv.org/abs/1904.01098>

源码: [UGraphEmb](#)

关键词: **Graph Embedding, Unsupervised Learning**

写于: 2020-10-14

该论文 [4] 与 27.2, 27.3 的作者相同, 这篇论文的目的是为了学习到图的表征, 并让学习到的表征尽量保持图之间的相近关系 (proximity)。学习到的表征可以作为下游任务的输入, 个人认为这相当于一种与训练。论文提出了一个无监督的模型 — UGraphEmb (Unsupervised Graph-level Embedding) 基于图之间的 proximity 来学习图表征, 并提出了多尺度 (类似于 [5] 中的多尺度) 的结点注意力 MSNA (Multi-Scale Node Attention)。

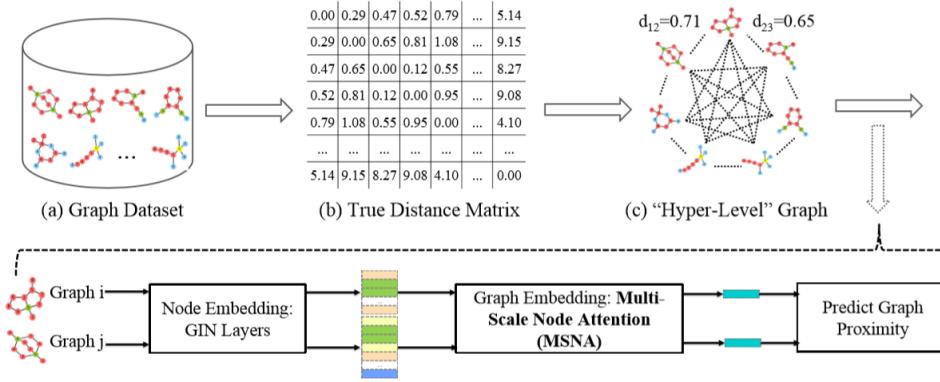


Figure 37: Overview of UGraphEmb

**UGraphEmb 思路** 首先, 这个模型是在一个图数据集上学习的, 需要先计算图数据集中两两图之间的 proximity (一开始我认为是 similarity, 但并不是很准确, 论文中使用的是 GED 来计算的 proximity), 这样就可以得到一个 distance matrix, 表示任意两个图之间的 proximity。那么如何生成图的表征呢? 使用 GIN[62] 来学习结点表征, 多尺度则体现在: 以 GIN 的每一层生成一个图表征, 再拼接在一起形成最终的图表征。其中每一层会有一个注意力, 也就是 MSNA。得到任意两个图的表征后, 就要使用 proximity 了, 希望模型学习到的图表征尽量保留图之间的 proximity。损失函数定义如下:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{(i,j) \sim \mathcal{D}} (\hat{d}_{ij} - d_{ij}) \\ &= \mathbb{E}_{(i,j) \sim \mathcal{D}} (\|\mathbf{h}_i - \mathbf{h}_j\|_2^2 - d_{ij})\end{aligned}$$

其中  $\hat{d}_{ij}$  是关于两个图表征的 proximity 计算,  $d_{ij}$  是预先计算好的 proximity。

论文中提到了图表征的无监督学习方法的一个挑战 : 与结点表征学习的不同, 结点之间存在联系, 可以在此基础上进行无监督学习, 而图之间是没有天然的联系的 (如距离/相似性)。而 proximity 就充当了一个这样的 **Inter-Graph information**, 将不同的图联系起来。

### 方法解决的问题/优势

- 无监督的、inductive 的图表征方法
- 可以作为图数据的预训练模型, 下游任务在其基础上进行微调即可, 如图分类、图相似性计算

## 方法的局限性/未来方向

- 能否直接在 Hyper-level graph 上学习图的表征与多尺度的表征融合呢?

## 27.5 Grapg-Graph Similarity Network

论文地址: <https://openreview.net/pdf?id=R3a2G2tSf3c>

关键词: Graph Similarity, Graph Classification, Metric Learning

写于: 2020-10-15

该论文针对图分类问题提出了新的解决方案 — Graph-Graph (G2G)。现有的图分类方法中，通常会忽略图数据之间的关系 — 当然图数据之间也没有直观的联系。论文中以图为结点，图之间的相似性为边的权重构建了 SuperGraph，之后图的分类问题则转变成 SuperGraph 中结点分类的问题。G2G 方法过程如 Fig.38 所示。

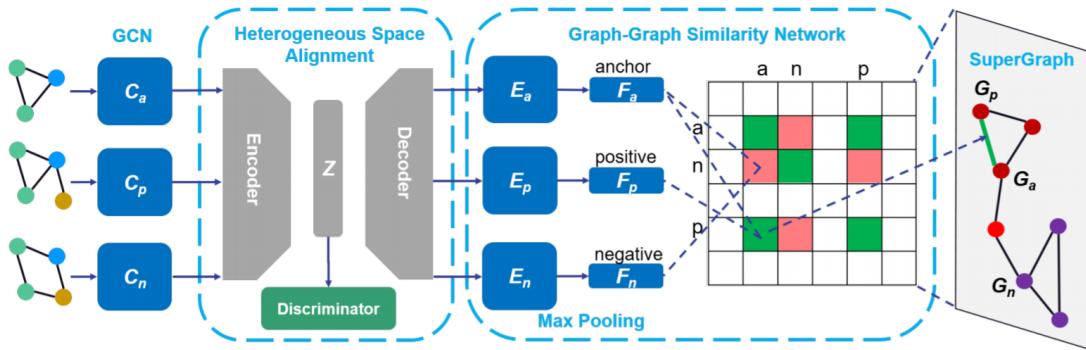


Figure 38: Overview of Graph-Graph

**G2G 思路** 如 Fig.38 所示，G2G 可以分为四个部分：

**GCN** 使用一个共享的 GCN 为所有的图的结点生成表征。

**Heterogeneous Space Alignment** 使用对抗自动编码器 [33] 用于对生成表征对齐到同一个空间。**这也是我的一个问题，不同的图中的结点的表征所在的空间之间有什么联系呢？**

**Graph-Graph Similarity Network** 先通过一个 Max pooling 来得到图的表征，再基于 Contrastive loss 和 Triplet loss，使用 NN 来学习图之间的相似性。

**SuperGraph** 基于图的表征和相似性矩阵构建一个 SuperGraph，每个图都是一个结点，相似性为边的权重。再图表征的基础上，可以使用传统的机器学习方法对 SuperGraph 中的结点 — 即原来的图进行分类，论文中使用的是 KNN 对结点进行分类。

有一点要注意，G2G 每次都使用 3 个图作为输入，分别是 anchor, positive, negative，anchor 和 positive 之间是同类的图，anchor 和 negative 之间是不同类的图，因此在训练过程中使用了 Contrastive loss 和 Triplet loss。

## 方法解决的问题/优势

- 在完成图分类任务的过程中，学习到了图之间的相似性
- 挖掘图的表征空间之间的关系，针对表征空间的问题提出了解决方案，将不同图的表征进行了对齐（**还有点不太理解文中的意思，原文是：align all the graphs to a same distribution**）
- 在对 graph-level 进行分类时，考虑了图之间的关系（similarity）
- 使用 triplet 进行寻来

## 方法的局限性/未来方向

- 在图类别较多的数据集上效果不是很好
- 现有的图分类方法中，主要考虑的是结点的特征和图的拓扑结构，实际中**边在图中也有很大的作用**，进行 graph-level 的任务时，能否考虑边的信息

虽然这篇论文还在双盲中，但我怀疑这篇论文又是Yunsheng Bai做的。

## 27.6 Deep Graph Similarity Learning: A Survey

论文地址：<https://arxiv.org/abs/1912.11615>

关键词：Metric learning, Graph similarity, GNN

写于：2020-10-17

该论文 [32] 对近年来使用深度学习/GNN 计算图相似性的方法进行了一个较为全面的总结，对每种方法进行了一个简要的介绍，描述了图相似性常用的数据集和评估方法，并对图相似性计算的应用及该领域内现存的挑战和难点进行了全面的概括。

图的相似性计算是一种 metric learning，学习如何度量两个对象之间的距离。图之间的相似性可以用一个 0 到 1 之间的小数表示，但是关键在于如何学习到这样的一个 similarity score。

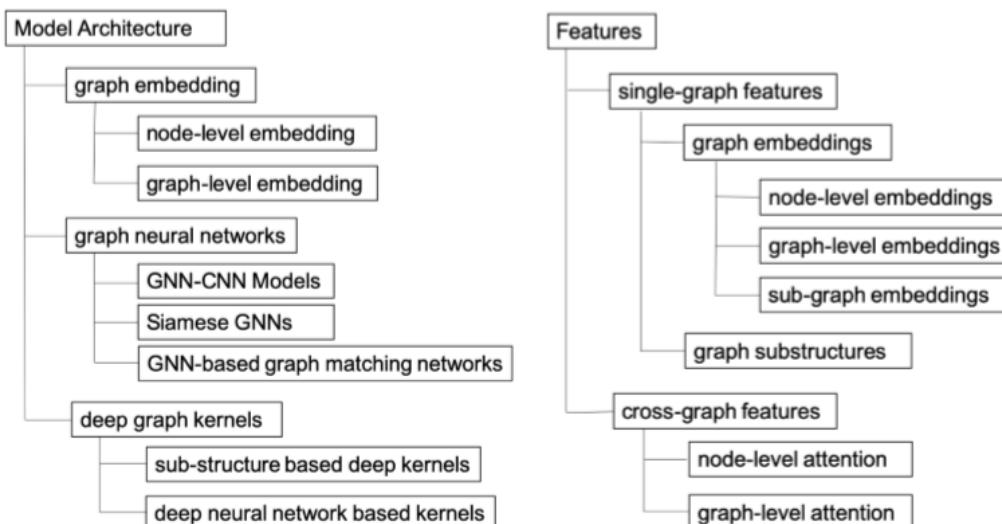


Figure 39: Taxonomy of DGS

如 Fig.39 所示，可以按照模型结构和单/多图的特征来进行分类。按照模型结构，论文中对现有的深度图相似性学习方法主要分为以下三类：

- graph-embedding based：使用通过 graph embedding 方法得到的结点/图表征来计算相似度，如 graph2vec, node2vec 等
- graph neural networks based：在图相似性的任务中，以 end-to-end 的方式使用 GNN 来学习相似度，如 GSimCNN, SimGNN, Siamese GCN 等
- deep graph kernel based：通过学习核函数来计算图之间的相似性，如 Deep graph kernels, Deep divergence graph kernel 等

论文中对图相似的应用归纳如下：

- 在生物信息领域内的应用。例如学习化学元素、分子、化合物等的相似性，研究它们之间的相互作用
- 在脑科学中，检测脑网络结构的相似性，分析脑网络的功能。可以指示脑网络功能的正常与否
- 计算机安全。可以用来对代码进行分析，
- 在计算机视觉中的应用，用于视频序列中动作行为的识别

论文中对图相似性的挑战归纳如下：

- 在复杂的图数据上的相似性计算。如有向图、动态图/流图、
- 解释学习到的结果。
- 小样本学习 (few-shot learning)。当每一类只有少量数据时如何学习相应的分类器

有几个注意的点：

- 一个图与自身的相似性要大于等于任何与其他图的相似性，即  $s_{ii} \geq s_{ij}$
- 数据的对齐直觉上是将不同空间/分布的数据放在同一个空间/分布中，但依然能够保持它们原有的某些性质（包括 intar, inter 的性质）
- 在相似性计算时利用结点/边的特征
- 由于脑网络的特殊性，脑网络中的相似性更应该是一个子图学习的问题

## 27.7 Hierarchical Large-scale Graph Similarity Computation via Graph Coarsening and Matching

论文地址：<https://arxiv.org/abs/2005.07115>

关键词：**Graph Similarity, GNN**

写于：2020-10-18

该论文 [61] 着眼于大规模图的相似性计算问题。生物/化学中分子识别的时先将分子映射到分子组再在分子组的基础上计算分子的相似性，受到这个启发，论文提出了对大规模图的相似性计算方法 — COSIM-GNN (Coarsen based Similarity Computation via Graph Neural Network)，按照“embedding-coarsening-matching”的过程计算图的相似性。

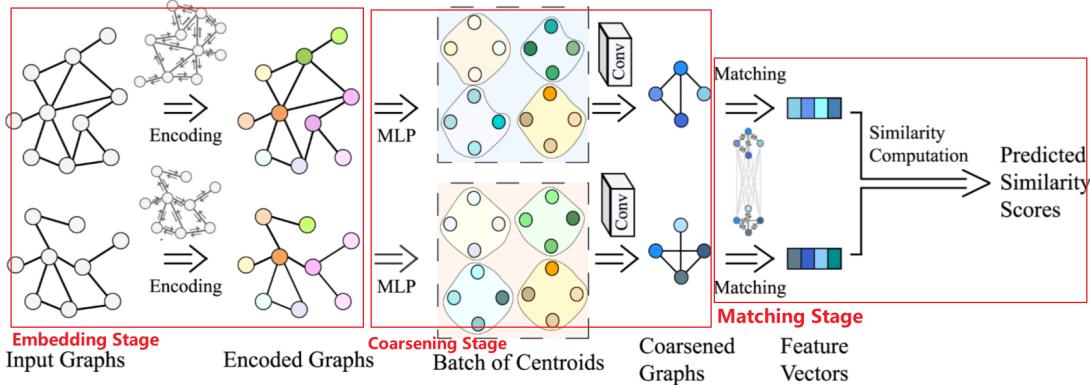


Figure 40: Overview of COSIM-GNN

**COSIM-GNN 思路** 整体过程如图 Fig.40 所示，可以分为 3 个阶段：

**Embedding** 对输入的两个图，使用 GNN 分别得到两个图的表征。

**Coarsening** 对于 Embedding 阶段得到的图结点表征，Coarsening 阶段，通过池化的方法得到一个粗化后的图。为了使得粗化后的图是和输入数据相关且是节点排列无关的，论文还提出了一个新的池化方法：Adaptive Pooling。该 pooling 方法在第一阶段的结点表征的基础上得到  $h$  个 batch — 每个 batch 表示一个不同的图，基于这些 batch 来获得最终压缩后的图。**为什么要获得  $h$  个 batch 呢？** — 为了尽量使得到的反映原图的压缩图。具体细节可参看论文（这一部分我没看太懂）。

**Matching** 之前的两个阶段中，是分别在输入的两个图上执行这两个阶段。每个图的结果，都只使用了图本身的信息，Matching 阶段则涉及到两个图之间的交互，在两个图的基础上一起计算相似性。具体可以使用一些匹配的方法来计算得到它们的相似性，如 GSimCNN

### 方法解决的问题/优势

- 针对大图的相似性计算给出了解决方法
- 利用了图之间的信息计算图的相似性，并且通过 coarsening 降低了这一过程的计算开销

### 方法的局限性/未来方向

- 改进粗化图的方法
- 改进图匹配的方法

## 27.8 Deep Graph Kernels

论文地址：<https://dl.acm.org/doi/pdf/10.1145/2783258.2783417>

来源：KDD’15, Sydney, NSW, Australia

作者：Pinar Yanardag, S.V.N. Vishwanathan

关键词：graph kernels, deep learning

写于：2020-11-02

该论文 [64] 针对传统的 graph kernel 方法的缺点提出了 Deep Graph Kernels ——一个能够学习图子结构的潜在表示的统一框架。

**使用 Graph Kernels 计算图相似性** 如下公式所示，其中  $\phi(\mathcal{G})$  表示图  $\mathcal{G}$  基于子结构的表示向量，例如该向量的每一维表示某一个子结构的出现次数； $\mathcal{M}$  表示了所有子结构之间的关系； $\mathcal{K}(G_1, G_2)$  是一个核函数，可以用来计算图之间的相似性。

$$\mathcal{K}(G_1, G_2) = \phi(G_1)\mathcal{M}\phi(G_2)$$

该论文的重要一点就是学习到子结构的向量表示并计算得到表示这子结构间关系的矩阵  $\mathcal{M}$ 。

传统的 graph kernel 方法是将图数据（不一定是图，也可以是其他类型的数据，如数、序列等）分解为许多小的子结构，在子结构的基础上就可以将一个图表示为基于子结构的向量。在这些子结构基础上定义有效的核函数（原始特征在高维空间中的内积），就可以使用核函数来计算图之间的相似性。但是传统的 graph kernel 方法有以下几个缺点：

- 通常来说，子结构之间并不是相互独立的。如 graphlets（拥有  $k$  个结点的，连通的非同构图）。 $k+1$  个结点的 graphlet 包含了  $k$  个结点的 graphlet
- 当图数据很大时，会有很多不同的子结构，图的基于子结构的向量表示将会变得很稀疏。只有一些子结构会出现在图中，在这种情况下，会出现 diagonal dominance ——一个图只与自己相似，与数据集中其他图不相似/相似度很低

个人觉得，该论文主要通过神经语言模型的方法学习到子结构的表示，再方便地计算  $\mathcal{M}$ 。但是论文中说的有点含糊。

简单的介绍以下 Graph kernels。有很多中 Graph kernel，比如基于 graphlets 的、基于 subtree 的、基于最短路径的等等，但除了如何生成子结构的方法不同，它们都会用子结构的频数向量来作为图的表示。graph kernels 的方法很像 language model 中的一些方法，如 word2vec 中的方法。

**思路** 论文针对不同的 Graph kernels 提出了一个统一的框架，能依据子结构之间的依赖性学习子结构的向量表示（即隐表示）。学习隐表示的方法借鉴了神经语言模型中的概念 — 使用连续的向量表示词使用上下文来定义 word。在该论文中，word 就是 graph kernel，每一个图可以视作 sentence，用多个 graph kernels 即可表示一个图。论文中针对不同的 graph kernels 方法，使用不同的方法将图分解为子结构的集合。但与语言模型不同的是，一个图生成的子结构间并没有一个明显的共线关系，因此论文中针对不同的 graph kernels 给出了不同的解决方法 — 使用不同的方法体现出子结构之间的共现性（也即子结构之间的依赖关系、上下文）。具体怎么产生子结构之间的共现性就不一样描述了，重点是根据 graph kernel 的特点，衍生出相关的子结构。在生成了子结构以其之间的共现关系后，就可以使用 CBOW/Skip-gram 算法来学习子结构的向量表示了。之后便可借助子结构的向量表示来计算图之间的相似性了。

## 方法解决的问题/优势

- 提出了一个统一的框架，能够在不同的 graph kernels 下学习到子结构的隐表示
- 利用 graph kernels 之间的依赖关系来定义子结构的共现性

## 方法的局限性/未来方向

- 对更复杂的图数据的 graph kernels 的定义和学习，如 attributed graphs
- graph kernels 方法主要关注的是图的结构信息，不适用于很多真实的图数据

## 28 Dynamic Graphs

### 28.1 EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs

论文地址: <https://arxiv.org/pdf/1902.10191>

来源: AAAI, 2020

作者: Pareja, Aldo, et al.

源码: EvolveGCN

关键词: Dynamic Graphs, GCN

写于: 2021-02-17

该论文 [43] 解决的是不断进化 (evolve) 的 graph 的表征问题。以往的方法主要将结点的表征通过 RNN 来学习不断进化的 graph, 这种方法在学习表征时需要知道结点在所有时间区间上的变化 (相当于是 transductive 的), 并且在变化比较频繁的 graph 上不太友好。论文提出 EvolveGCN, 将 GCN 和 RNN 结合, 使用 RNN 来学习 GCN 的参数, 以此来捕捉动态图的动态性。

**问题定义** 对于一个动态图  $G$ , 在每一个时间点  $t$  上, 可以将  $G$  表示为  $(A_t, X_t)$ , 分别表示时间  $t$  时的邻接矩阵和特征矩阵。最终要学习的是  $G$  中每个结点在各个时间点上的结点表征。

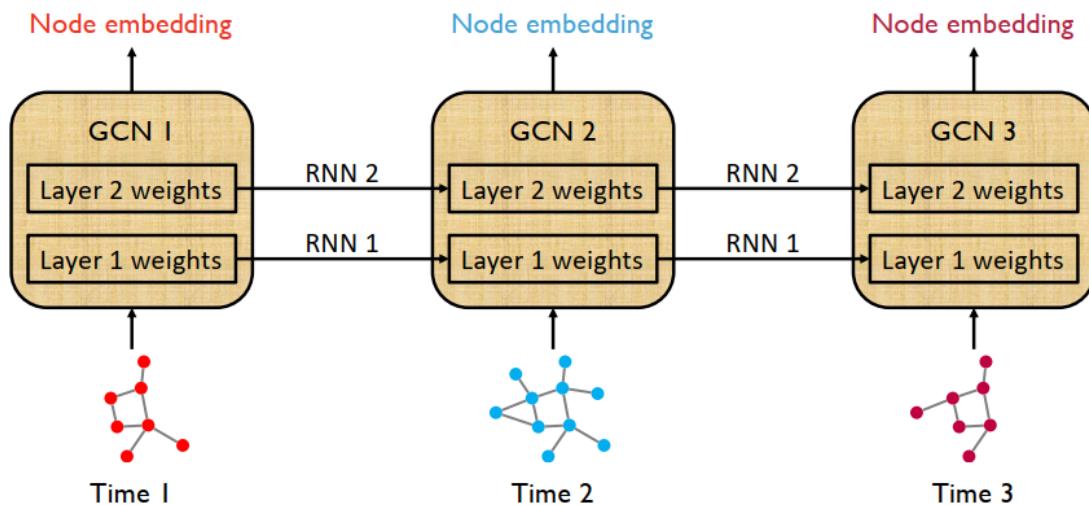


Figure 41: EvolveGCN

**EvolveGCN 思路** 如 Fig.41 所示, 在时间点  $t$  时, 将  $(A_t, X_t)$  输入到模型中, 可以学习到  $t$  时的结点表征。注意, **GCN 的参数并不是训练得来的, 而是通过 RNN 计算得到的**。GCN 在 EvolveGCN 中起的作用: 通过  $(A_t, X_t)$  得到结点表征, 但是并不会在计算表征的过程中更新 GCN 各层的参数。RNN 在 EvolveGCN 中起的作用: 在  $t - 1$  时的结点表征和 GCN 参数的基础上更新 GCN 的参数, 更新公式如下:

$$\underbrace{W_t^{(l)}}_{\text{hidden state}} = \text{GRU}(\underbrace{H_t^{(l)}}_{\text{input}}, \underbrace{W_{t-1}^{(l)}}_{\text{hidden state}})$$

或

$$\underbrace{W_t^{(l)}}_{\text{output}} = \text{LSTM}(\underbrace{W_{t-1}^{(l)}}_{\text{input}})$$

从上可以得到，动态图的变化都保存在了 GCN 的参数中。在实现时，需要对 RNN 进行更改：

- 将 RNN 单元的输入与隐状态扩展为矩阵形式，因为 GCN 的参数和结点表征都是矩阵的形式
- 将输入的列与隐状态的列相匹配

### 方法解决的问题/优势

- 对动态图的表征学习提出了可行的解决方法
- 将 RNN 和 GCN 相结合，利用 RNN 来进化 GCN 的参数，能够应对变化频繁的 graph，而且不需要提前预知结点的所有变化

### 方法的局限性/未来方向

- 虽然不需要提前预知结点的所有变化，但是**需要预知 graph 中的所有结点，不能应对结点的变化**
- 使用 RNN 来进化 GCN 的参数，将 graph 的动态性保存在参数中，一个训练好的模型也许只能捕捉一种动态性

## 28.2 Temporal Graph Networks for Deep Learning on Dynamic Graphs

论文地址：<https://arxiv.org/pdf/2006.10637.pdf>

来源：ICML 2020 Workshop

作者：Emanuele Rossi, et al.

源码：[tgn](#)

slides：[TGN: Temporal Graph Networks for Dynamic Graphs](#)

关键词：**graph representation, temporal graph**

写于：2021-03-01

该论文 [47] 主要解决的是连续时间下的动态图的表示，论文中将连续时间下的动态图看成流图 — 由一系列的带时间戳的事件组成。

**问题定义** 论文中将动态图建模为带时间戳的事件序列： $\mathcal{G} = \{x(t_1), x(t_2), \dots\}$ 。事件  $x(t)$  有两种类型：1) 结点事件，新增一个结点或者更新一个结点的特征，用  $\mathbf{v}_i(t)$  表示；2) 交互事件：表示结点之间的交互，由  $\mathbf{e}_{ij}(t)$  表示。该论文的主要贡献就是建立了一个通用的框架来对动态图进行建模，并且将一系列的模块组合起来对动态图进行学习，得到结点在某个时间点上的表征。**将 TGN 作用在一个连续时间的动态图上，将为每个结点在每个时间点上产生一个表征。**

**TGN 思路** 简言之，TGN 可以看作一个编码器，共由 5 个子模块组成。

**Memory**  $t$  时的模型的 Memory 由模型到  $t$  时为止所见到的结点  $i$  的状态  $s_i(t)$  组成。结点的状态相当于对结点的历史信息的一个压缩形式的表示。

**Message Function**  $t$  时, 当有涉及  $i$  结点的事件时, 一条 message 会被计算用于更新  $i$  的状态。论文中对两种事件的消息计算公式如下:

$$\mathbf{m}_i(t) = msg_s(\mathbf{s}_i(t^-), \mathbf{s}_j(t^-), \Delta t, \mathbf{e}_{ij}(t))$$

$$\mathbf{m}_j(t) = msg_d(\mathbf{s}_j(t^-), \mathbf{s}_i(t^-), \Delta t, \mathbf{e}_{ij}(t))$$

$$\mathbf{m}_i(t) = msg_n(\mathbf{s}_i(t^-), t, \mathbf{v}_i(t))$$

$msg_s, msg_d, msg_n$  分别表示交互事件中源节点和目标节点、结点事件中所涉及的结点的消息的计算公式。具体实现时可以有多种形式, 如 MLP, 在论文中采用的是拼接的形式。

**Message Aggregator** 由于在实际中通常会采用一次性处理多个事件 (可能是多个时间的多个事件), 需要将多个事件对应的 message 汇聚, 形式如下:

$$\bar{\mathbf{m}}_i(t) = agg(\mathbf{m}_i(t_1), \dots, \mathbf{m}_i(t_b))$$

$agg$  的具体实现形式也可以是多样的, 如 RNN、mean 等, 论文中采用的是只选择最近的 message。

**Memory Updater** 得到了  $t$  时设计到  $i$  的 message 后就可以更新状态了。

$$\mathbf{s}_i(t) = mem(\bar{\mathbf{m}}_i(t), \mathbf{s}_i(t^-))$$

同样,  $mem$  的形式也可以是多样的, 如 LSTM、GRU 等。

**Embeddings** 这个模块是用于为  $i$  生成在  $t$  时的表征的:

$$\mathbf{z}_i(t) = emb(i, t) = \sum_{j \in N_i^k([0, t])} h(\mathbf{s}_i(t), \mathbf{s}_j(t), \mathbf{e}_{ij}, \mathbf{v}_i(t), \mathbf{v}_j(t))$$

$h$  的具体形式也是多样的。

最终 TGN 的整体如 Fig.42 所示。

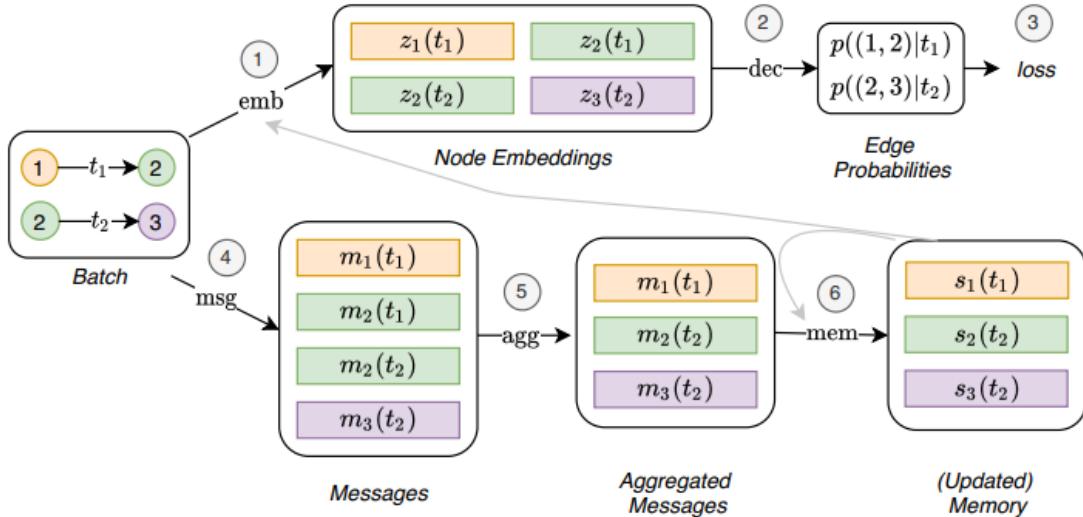


Figure 42: TGN

## 方法解决的问题/优势

- 对连续时间下的动态图进行了建模，以事件序列表示一个动态图
- 对动态图的表征学习提出了一个统一的框架
- 可应用于异常检测、金融交易等场景中

### 28.3 Inductive representation learning on Temporal Graphs

论文地址: <https://openreview.net/pdf?id=rJeW1yHYwH>

来源: ICLR, 2020

作者: Da Xu, Chuanwei Ruan, et al.

源码: [TGAT](#)

关键词: self-attention mechanism, graph representation, temporal graph

写于: 2021-03-01

该论文 [12] 也是为了解决动态图的表征问题。改论文在 self-attention[55] 机制的基础上，提出了 temporal graph attention(TGAT) layer 来汇集 temporal-topological neighborhood features。

**问题定义** 该论文讨论的也是连续时间下的动态图的表征问题，与 [47] 不同，该论文并没有将动态图用事件序列来表示，重点关注的时图中结点的 temporal neighborhoods，如何利用 temporal neighborhoods 来计算结点在  $t$  时的表征。

**TGAT 思路** 在介绍 TGAT 之前，先来了解以下 self-attention 是啥。

**self-attention** 简单的来说自注意力机制就是输入一个序列的数据，以序列自身来计算序列中各个元素与其他元素（包括自身）之间的关系大小（注意力大小）。self-attention 最开始用于自然语言处理中。在计算自注意力时需要将位置编码与序列中元素的表征相结合。

$$Attn(Q, K, V) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$$

上式中的  $\mathbf{Q}, \mathbf{K}, \mathbf{V}$  都是由同一个矩阵  $\mathbf{X}$  (表征矩阵) 乘以不同的权重矩阵得到的。

**TGAT** TGAT 中将 self-attention 中的 positional encoding 替换为 time encoding，再将 time encoding 与结点特征相结合构成 Temporal Graph Attention Layer。time encoding 的输入为时间戳，输出为一个向量。对于时间  $t$ ，结点  $i$  的 temporal neighborhoods 包括在时间  $t$  之与  $i$  发生了交互（即存在过边）的结点，每个结点上一层的表征和时间表征拼接起来作为 self-attention 中的  $\mathbf{X}$ ，即可计算得到当前节点与其 temporal neighborhoods 之间的注意力，邻居结点的上一层的表征加权求和后隐表示，该隐表示再与结点的特征向量拼接后输入到 MLP 中，最终的输出为节点在  $t$  时的 TGAT 层输出。TGAT layer 如 Fig.43 所示。

说实话，TAGT 的大概意思看懂了，但是其中涉及 time encoding 转化为一个分布学习的问题、TGAT layer 的具体实现过程不是很懂。

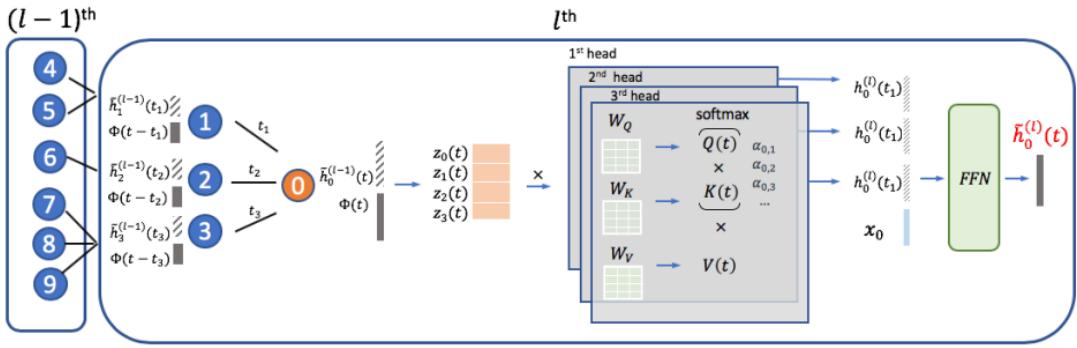


Figure 43: TGAT layer, 采用了多头注意力机制, k=3

### 方法解决的问题/优势

- 引入了 temporal 的图注意力机制
- 将 self-attention 中的 positional encoding 替换为 time encoding
- 引入了注意力机制能够增加模型的可解释性

### 方法的局限性/未来方向

- 对于  $t$  时的结点的表征, 需要考虑  $t$  之前的所有相关邻居, 如果  $t$  很大时可能会存在计算开销很大的问题
- 结点的特征已经在第一层输入了, 但是在 TGAT 层中还将节点特征与隐表征拼接后输入到 FFN 中, 这样是否有必要?

## 28.4 DySAT: Deep Neural Representation Learning on Dynamic Graph via Self-Attention Networks

论文地址: [http://yhwu.me/publications/dysat\\_wsdm20.pdf](http://yhwu.me/publications/dysat_wsdm20.pdf)

来源: WSDM, 2020

作者: Aravind Sankar, Yanhong Wu, et al.

源码: [DySAT](#)

关键词: **self-attention, representation learning, dynamic graphs**

写于: 2021-03-02

该论文 [49] 解决的是动态图中的结点表征问题。论文提出了 DySAT (Dynamic Self-Attention), 以自注意力机制捕捉动态图的结构的动态性。DySAT 分别从两个方面捕捉动态性: structural neighborhoods 和 temporal dynamics, 并且使用多头注意力来捕捉多方面的动态性。

**问题定义** 论文中使用图序列来表示动态图。关于动态图的建模, 在这里插一句。

**dynamic graph** 动态图通常由两种建模方式: 图序列 (graph snapshots) 和基于带时间戳的事件的图 (time stamped events, 类似于流图)。本质上来看这两种建模方式是等价的, 是可以互相转化的。但是不同的建模形式针对这不同的应用场景。snapshot 形式的动态图, 直观上强调的整体性, 图中结点/边的变化是为了整体的图而服务的, 这种情况下我们更多的考虑的是作为一个整体的图的应用场景, 例如在场景识别中、对图进行分

类的任务中。而 timestamped 形式的动态图，对整体的考虑可能会不是那么强，更强调的是图中结点/边以及这些变化对任务的影响。

作者采用的是 snapshots 形式的动态图  $\mathbb{G} = \{\mathcal{G}^1, \dots, \mathcal{G}^T\}$ 。其中  $T$  是时间步的数量， $\mathcal{G}^t = (\mathcal{V}, \mathcal{E}^t)$ 。显然，该论文中动态图的结点是不变的，即不涉及结点的增加或删除。最终的目标就是学习图中每个结点在任意时间  $t$  时的表征。

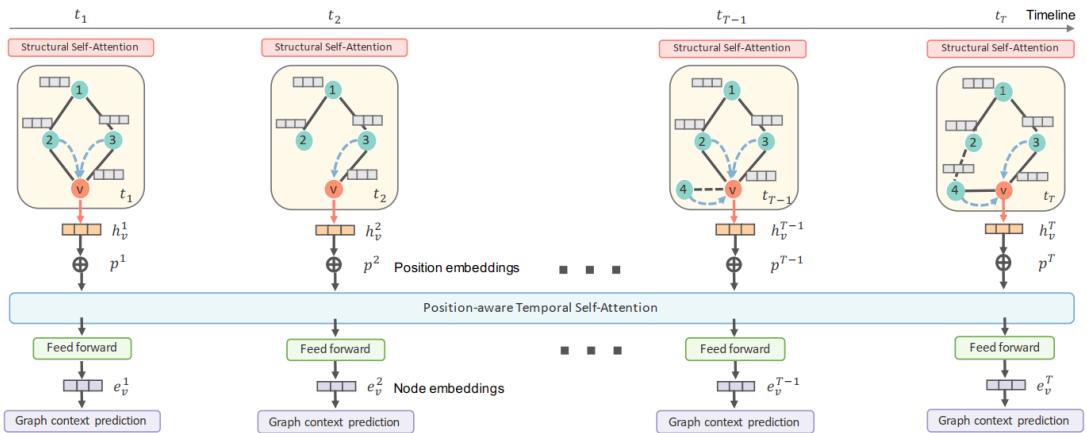


Figure 44: DySAT

**DySAT 思路** DySAT 的整体框架如 Fig.44 所示。DySAT 主要分为两个部分：Structural Self-Attention 和 Temporal Self-Attention。

**Structural Self-Attention** 这一部分与 GAT 中的注意力机制类似，相当于一个邻居结点信息汇聚层。对于每一个 snapshot graph，Structural Self-Attention 利用当前时刻各个结点的表征计算注意力再进行加权求和，计算公式如下：

$$z_v = \sigma \left( \sum_{u \in \mathcal{N}_v} \alpha_{uv} W^s x_u \right), \alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{w \in \mathcal{N}_v} \exp(e_{vw})}$$

$$e_{uv} = \sigma \left( A_{uv} \cdot a^T [W^s x_u \| W^s x_v] \right) \forall (u, v) \in \mathcal{E}$$

**Temporal Self-Attention** 这部分是为了捕捉动态图在时间上的变化模式。计算结点  $v$  在  $t$  时的表征时，将在  $t$  之前的  $v$  的表征作为 temporal self-attention 模块的输入，输出的是结点  $v$  在各个事件点的表征（此时的表征考虑了动态性），计算公式如下：

$$Z_v = \beta_v (X_v W_v), \quad \beta_v^{ij} = \frac{\exp(e_v^{ij})}{\sum_{k=1}^T \exp(e_v^{ik})}$$

$$e_v^{ij} = \left( \frac{\left( (X_v W_q) (X_v W_k)^T \right)_{ij}}{\sqrt{F'}} + M_{ij} \right)$$

上式的形式与 self-attention 的形式一致。其中的  $M \in \mathbb{R}^{T \times T}$  是一个掩码矩阵，

$$M_{ij} = \begin{cases} 0, & i \leq j \\ -\infty, & \text{otherwise} \end{cases}$$

通常一个注意力捕捉的是一个方面的性质，为了捕捉动态图中多个方面的动态性，作者引入了多头注意力，Structural 和 Temporal 都引入了多头注意力机制。

为了训练模型中的参数，论文使用类似神经语言模型中的共现率来优化参数，这点与 word2vec 和 Node2vec

中的损失函数很像，损失函数如下， $P_n^t$  为负采样的结点：

$$L = \sum_{t=1}^T \sum_{v \in \mathcal{V}} \left( \sum_{u \in \mathcal{N}_{\text{walk}}^t(v)} -\log(\sigma(\langle e_u^t, e_v^t \rangle)) - w_n \cdot \sum_{u' \in P_n^t(v)} \log(1 - \sigma(\langle e_{u'}^t, e_v^t \rangle)) \right)$$

DySAT 是先进行 structural self-attention 再进行 temporal self-attention，作者这样设计是因为：随着时间变化，图的结构是不稳定的。

### 方法解决的问题/优势

- 提出了 structural 和 temporal self-attention 来学习动态图中的结点表征
- 使用多头注意力机制捕捉多方面的动态性
- 注意力机制适用于并行

### 方法的局限性/未来方向

- 只能适用于结点不变化的动态图
- 以结点的共现率为损失函数引导模型的训练，这样的损失函数可能重点关注的是图的结构，对于动态性更丰富的图（如结点的特征的变化）是不够的

## 28.5 Spatial Temporal Graph Convolutional Network for Skeleton-Based Action Recognition

论文地址：<https://arxiv.org/pdf/1801.07455.pdf>

来源：AAAI, 2018

作者：Sijie Yan, Yuanjun Xiong, Dahua Lin

源码：[st-gcn](#)

关键词：Action Recognition, GCN

写于：2021-03-04

$$\int_{\partial \text{hourglass}} \text{frog} = \int_{\text{hourglass}} d\text{frog}$$

该论文 [63] 针对人体动作识别问题提出了解决方案。以 Skeleton 为基础构建时空图，提出了 ST-GCN (Spatial Temporal GCN) 对人体行为的时空图进行表征，在进行行为分类。

**问题定义** 给定一段视频，确定其中的人体行为类别。可以从多个方面表示人体行为，如光流、骨骼关键点等，该论文中采用基于骨骼关键点的形式表示人体行为。在一段视频中，每一帧中的人体骨骼点可以构成一个 graph — Spatial graph，帧与帧之间同一个骨骼点进行连接可以构成 graph — Temporal graph。论文的目标就是基于一个视频的骨骼点构成的 Spatial Temporal graph 来对人体行为进行分类。

### ST-GCN 思路

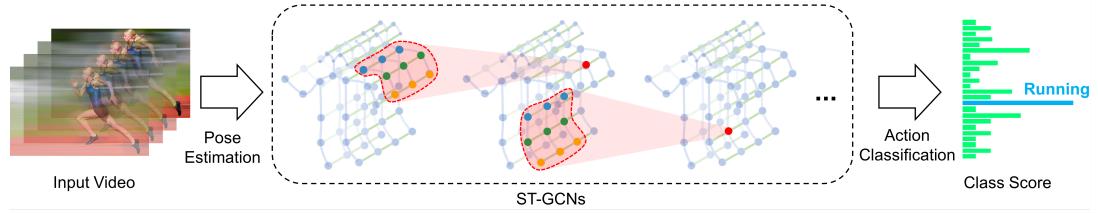


Figure 45: ST-GCN

**Skeleton Graph Construction** Spatial Temporal Graph  $G = (V, E)$ ,  $V = \{v_{ti} | t = 1, \dots, T, i = i, \dots, N\}$  中的结点来自人体骨骼关键点,  $N$  表示关键点个数,  $T$  表示视频帧数。每一帧中的 Graph 构建以人体自然连接为依据, 帧与帧之间的 Graph 构建方式为: 相邻帧之间, 同一个关键点进行相连。因此, 边也分为两种:  $E_S = \{v_{ti}v_{tj} | (i, j) \in H\}$ ,  $E_F = \{v_{ti}v_{(t+1)i}\}$ , 其中  $H$  表示人体中关键点的自然连接。

**Spatial Graph Convolutional Neural Network** 作者在 Spatial Temporal Graph 的基础上定义了 Spatial Temporal Convolution。这个卷积操作与 GCN 中的卷积很类似, 但是加上了一个时间维度 — 每个结点的邻居不仅要考虑当前帧中相邻的结点还要考虑时间上相邻帧的连接。对于图中每个结点, 其特征为骨骼关键点在帧中的位置。

经过多层的 Spatial Temporal Convolution 后, 将特征输入到 MLP 中进行行为分类即可。

### 方法解决的问题/优势

- 首次将 GCN 应用于行为识别
- 不仅考虑了帧中各个关键点, 还考虑了帧间关键点

### 方法的局限性/未来方向

- 论文中提到的注意力机制没有明显的意义
- 只考虑了局部特征

## 28.6 Inductive Representation Learning In Temporal Networks via Causal Anonymous Walks

论文地址: <https://arxiv.org/pdf/2101.05974.pdf>

来源: ICLR, 2021

作者: Yanbang Wang, et al.

源码: [CAW](#)

关键词: **Link Prediction, Causal Anonymous Walk, Temporal Graph**

写于: 2021-03-09

该论文 [58] 就动态网络的模式抽取问题, 针对现有方法中的一些弊端: 如主要依靠结点 id 和丰富的边的属性且难以抽取复杂的模式, 提出了 CAW (Causal Anonymous Walk) 以 inductive 的形式表示动态图。CAW 也是一种游走, 在 Temporal networks 中以 anonymization 的策略游走生成 CAWs, 将这些 CAWs 作为 motifs 来表示动态网络。CAWs 不需要传统的基于核方法中的选择什么样的 motifs、统计 motifs 个数不同。再者, 论文中提出了神经网络模型 CAW-N 用于编码 CAWs。

**问题定义** 动态网络中存在众多的模式，例如三角闭包以及一些更复杂的模式，目标就是在动态网络的表示中能够包含这些复杂的模式。当前对与动态网络的表示只能表示一些简单的模式或者主要停留在静态网络上。对于动态网络的研究主要由三个挑战：1) 动态网络中结构和时序的混合，需要一个更好的模型对动态网络进行建模；2) 模型的伸缩性，动态网络会不断地被新到来的数据更新；3) 能够学习训练数据中出现过的数据，并提取出它的模式。

这里介绍一下 inductive 和 transductive：Transduction is reasoning from observed, specific (training) cases to specific (test) cases. In contrast, induction is reasoning from observed training cases to general rules, which are then applied to the test cases.

论文中用  $\mathcal{E} = \{(e_1, t_1), (e_2, t_2), \dots\}$  表示动态网络，其中的含义显然易见。这个序列的边就编码的网络的动态性，那么，**动态网络的表示模型的表示能力就取决于能否准确地基于以往的信息预测结点之间的连接**。论文中也使用链接预测来作为模型的指标。

再介绍一些用到的符号： $\mathcal{E}_{v,t} = \{(e, t') \in \mathcal{E} \mid t' < t, v \in e\}$  表示在  $t$  之前与  $v$  相连的边集；下式表示一个 CAW，注意其中时间使从大到小的，及新的边放在前面， $W[i]$  表示第  $i$  个结点-时间对。

$$W = ((w_0, t_0), (w_1, t_1), \dots, (w_m, t_m)), t_0 > t_1 > \dots > t_m, (\{w_{i-1}, w_i\}, t_i) \in \mathcal{E} \text{ for all } i$$

**CAW 思路** 在 CAW 之前有 Anonymous walks (AW) [38]，如 Fig.46 所示，AW 很像随机游走，但是又与之有点不同——用结点在 walks 中出现的顺序替代结点的 id。这里也给我一个较大的启示：**实践中对结点的编号并不能体现 graph 中的模式，在学习模式的表征时，应尽可能将实践时引入的编号等一些辅助信息与 graph 的结构、模式等区分开来。但是在实践时有时需要一个明确的表示的，可以借助一个中间的表示，将在不同应用场景下的表示转化为中间表示，用中间表示来表示模式。这有点类似于虚拟机和神经机器翻译中的概念**。

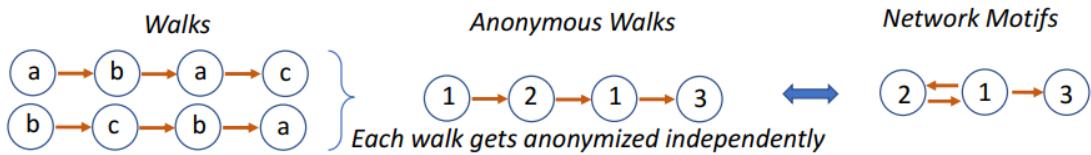


Figure 46: Anonymous walks

CAW 中就采用了这种思想，将结点的编号用结点在某个特定位置出现的次数代替。对于给定的两个结点，当预测它们的边时，需要先分别以这两个结点为起点抽取 CAWs，再用论文中的方法替换结点的编号，再分别对 walks 进行编码，最后在预测边的概率。论文中提出的 CAW-N 主要由以下这些部分组成。

**Causal Anonymous Walk** Causal Anonymous Walk 由两部分组成：Causality Extraction 和 Set-based Anonymization，如 Fig.47 所示。

Causality Extraction 这一部分主要用于从动态网络中抽取出 walks（这个时候还不能称为 Causal Anonymous Walks），抽取算法如 Fig.48 所示。对于结点  $u_0, v_0$ ，该算法会分别以  $u_0, v_0$  为起点得到长度为  $m$  的  $M$  条 CAWs，分别为  $S_u, S_v$ 。得到 walks 后，需要将中的结点 id 去掉，转化成“中间表示”。转换过程就由 Set-based Anonymization 完成。

这一部分就是将 walks 中的结点 id  $w$  转化为相对结点 id  $I_{CAW}(w; \{S_u, S_v\})$ 。[38] 中只是基于一条路径进行转换，这是基于一个假设：任意两个 AWs 不会共享结点 id。显然，该论文中的转换是基于  $2M$  条 CAWs 转换的，这也是基于一个假设：**CAWs 之间的关系可能是动态图中模式的关键之处**。因此共享结点 id 的 CAWs 就相当于保留了这种关系。 $I_{CAW}(w; \{S_u, S_v\})$  定义如下。

$$I_{CAW}(w; \{S_u, S_v\}) \triangleq \{g(w, S_u), g(w, S_v)\}$$

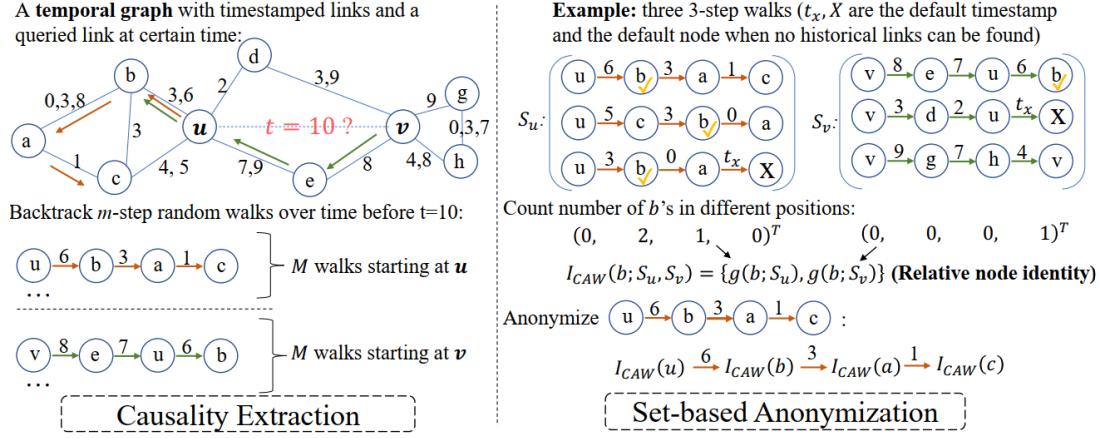


Figure 47: CAW

---

### Algorithm 1: Temporal Walk Extraction ( $\mathcal{E}, \alpha, M, m, w_0, t_0$ )

---

```

1 Initialize  $M$  walks:  $W_i \leftarrow ((w_0, t_0)), 1 \leq i \leq M$  ;
2 for  $j$  from 1 to  $m$  do
3   for  $i$  from 1 to  $M$  do
4      $(w_p, t_p) \leftarrow$  the last (node, time) pair in  $W_i$ ;
5     Sample one  $(e, t) \in \mathcal{E}_{w_p, t_p}$  with prob.  $\propto \exp(\alpha(t - t_p))$ 
      Denote  $e = \{w', w\}$  and then  $W_i \leftarrow W_i \oplus (w', t)$ ;
6 Return  $\{W_i | 1 \leq i \leq M\}$ ;

```

---

Figure 48: Temporal Walk Extraction

对于  $w_0 \in \{u, v\}$ ,  $g(w, s_{w_0}) \in \mathbb{Z}^{m+1}$  表示  $w$  在集合  $S_{w_0}$  中各个位置出现的次数。于是对于每一个 walk 可以表示为:

$$\hat{W} = ((I_{CAW}(w_0), t_0), (I_{CAW}(w_1), t_1), \dots, (I_{CAW}(w_m), t_m))$$

**Neural Encoding for Causal Anonymous Walks** 这一步是利用神经网络模型对  $\hat{W}$  进行编码，再将所有编码后的  $\hat{W} \in S_u \cup S_v$  聚集起来。

**Encode  $\hat{W}$**  文章采样的方法是将 walk 中的每个元素先进行编码，再输入到序列模型中，如下：

$$\text{enc}(\hat{W}) = \text{RNN} \left( \{f_1(I_{CAW}(w_i)) \oplus f_2(t_{i-1} - t_i)\}_{i=0,1,\dots,m} \right), \text{ where } t_{-1} = t_0$$

其中  $f_1, f_2$  分别是  $I_{CAW}(w_i), t_{i-1} - t_i$  的编码函数， $f_2$  就相当于 [12] 中的 time encoding。 $f_1$  的实现可以是一个 MLP，如  $f_1(I_{CAW}(w_i)) = \text{MLP}(g(w_i, S_u)) + \text{MLP}(g(w_i, S_v))$ 。

**Encode  $S_u \cup S_v$**  这一步就是聚集各个 walk 的 encoding 作为  $S_u \cup S_v$  的 encoding 来进预测  $u, v$  之间边的概率。论文中给出了两种参考的聚集方式：

- Mean-AGG( $S_u \cup S_v$ ):  $\frac{1}{2M} \sum_{i=1}^{2M} \text{enc}(\hat{W})$
- Self-Att-AGG( $S_u \cup S_v$ ):  $\frac{1}{2M} \sum_{i=1}^{2M} \text{softmax}(\{\text{enc}(\hat{W}_i^T) Q_1 \text{enc}(\hat{W}_j)\}_{1 \leq j \leq n}) \text{enc}(\hat{W}_i) Q_2$ , 其中  $Q_1, Q_2 \in \mathbb{R}^{d \times d}$

再将  $S_u \cup S_v$  的 encoding 输入到 MLP 中进行预测。

论文中还提到了如何扩展模型使之能够融入结点/边的属性，将属性拼接到  $\text{enc}(\hat{W})$  的输入即可。

## 方法解决的问题/优势

- 提出了一个新的动态网络的表征模型 CAW-N，能够对 temporal network motifs 进行编码，且该模型是 inductive 的
- 提出了一种新的方法来将 walks 转换到一个中间的表示空间
- CAWs 不仅保留了网络中的 temporal-spatial 结构，还保留了 motifs 之间的关系，能够捕捉更复杂的模式

## 方法的局限性/未来方向

- 论文中只针对连接预测进行了实验，不知道在其他任务如结点分类等任务上效果怎么样
- 实验中对 walk 的长度取的都比较小，这样提取到的模式可能是比较短期的，较简单的模式

## 28.7 DISCRETE GRAPH STRUCTURE LEARNING FOR FORECASTING MULTIPLE TIME SERIES

论文地址：<https://arxiv.org/pdf/2101.06861.pdf>

来源：ICLR, 2021

作者：Chao Shang, Jie Chen, Jinbo Bi

源码：[GTS](#)

关键词：Time series forecasting, graph structure learning

写于：2021-06-15

该论文 [50] 针对的是时间序列预测问题，要做到多步（multiple time）预测。论文将时间序列预测与 GNN 联合起来，以  $T$  步长的历史数据预测未来  $\tau$  步。

**问题定义**  $\hat{X}_{t+T+1:t+T+\tau} = f(A, w, X_{t+1:t+T})$  其中  $X$  是时间序列数据， $X.shape = [n\_series, steps, feature]$ 。  
 $X_t$  表示所有时间序列在  $step = t$  时的特征。现有的一些论文表明利用样本之间的关系能够辅助时间序列预测。因此，论文中引入了 Graph，来表示时间序列样本之间的关系。重点是如何生成这样的一个 graph。

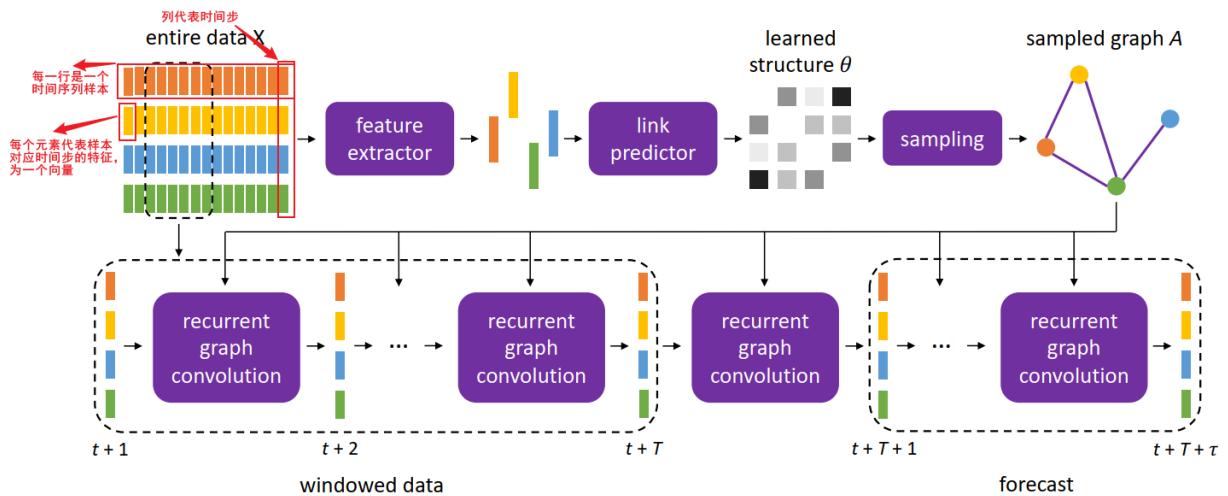


Figure 49: GTS Architecture

**GTS 思路** GTS (Graph for time series) 框架如 Fig.49 所示。看一眼损失函数：

$$\sum_t \ell(f(A, w, X_{t+1:t+T}), X_{t+T+1:t+T+\tau})$$

很明显，在训练过程中，类似于用滑动窗口在时间序列样本的时间方向上滑动，再计算损失。具体的计算公式不用纠结。上文中提到了，要构建一个 graph，那怎么构建呢？

对于  $n$  个训练样本，可以构建一个  $n$  个结点的 graph，剩下的就是邻接矩阵了。论文中将邻接矩阵中每个元素（论文中构建的是有向图）的取值服从于一个伯努利分布  $\theta$ 。 $\theta$  也是训练过程中学习得参数，因为邻接矩阵是从  $\theta$  中采样出来的，因此文中使用了 Gumbel[21] 重参数化技术。模型的前向过程从 Fig.49 可以体现。

**总结** 从模型的输入可以看出，模型针对的数据主要是样本数不变的场景，如交通预测中，而且每个样本的序列长度是一致的。可以看作是针对一个固定结点集的 graph，预测其结点特征。可以看作是一个回归任务，文中使用的也是均方误差作为损失。

## 29 Recommender System

### 29.1 Factorization Machines

论文地址：<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5694074>

来源：ICDM, 2010

作者：Steffen Rendle

源码：[libFM](#)

关键词：sparse data, tensor factorization, support vector machine

写于：2021-08-10

该论文 [45] 提出了一种新的模型 — Factorization Machines(FM)，本质上是一个类似于 SVM, LR 的实值特征预测函数，但是 FM 能够应对及其稀疏的数据（如推荐场景）、包含了特征间的交互。一些关键点：

- combines the advantages of Support Vector Machines (SVM) with factorization models
- FMs are a general predictor working with any real valued feature vector
- FMs model all interactions between variables using factorized parameters

#### 问题定义

$$y : \mathbb{R}^n \longrightarrow T$$

$T$  是目标域，可以是实数，也可以是  $\{+, -\}$ ，其中  $\mathbb{R}^n$  可能是及其稀疏的。

**FM** 先上公式：

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

模型的输入就是  $\mathbf{x} \in \mathbb{R}^n$ ，线性部分的参数  $\mathbf{w} \in \mathbb{R}^n$ ，二阶特征交叉的权重是  $\mathbf{V} \in \mathbb{R}^{n \times k}$ 。 $\mathbf{v}_i$  表示输入向量中第  $i$  个特征的向量表示。

如果写成这样：

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n w_{ij} x_i x_j$$

那么交叉特征的权重组成的矩阵  $\mathbf{W} = [w]_{ij}$  将是一个对称矩阵，而且  $\mathbf{W} = \mathbf{V}\mathbf{V}^T$  (正定矩阵可以通过 Cholesky 矩阵分解方法分解为一个元素非负下三角与其转置的乘，但**怎么保证  $\mathbf{W}$  是正定的呢？**)。那么为什么要写出一开始那个形式呢？

如果交叉特征的权重写成对称矩阵  $\mathbf{W}$  的形式，那么对于特征  $i, j$  来说，要学习到  $w_{ij}$  则需要足够的样本满足一定条件： $x_i! = 0 \cap x_j! = 0$ ，当  $x_i = 0 \cup x_j = 0$  则对更新  $w_{ij}$  没有帮助。

而把交叉特征的权重写成  $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$  的形式则不同了，由于每个特征都有一个隐向量  $\mathbf{v}_i$ ， $\mathbf{v}_i$  的更新可以通过  $x_i, x_j$  更新，也可以通过  $x_i, x_k$  来更新，当然，要更新  $\mathbf{v}_i$ ，对特征  $x_i$  是有要求，有什么要求呢？

先来对 FM 的公式的交叉特征部分进行一个化简：

$$\sum_{i=1}^n \sum_{j=i+1}^n \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \quad (2)$$

$$= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j - \frac{1}{2} \sum_{i=1}^n \langle \mathbf{v}_i, \mathbf{v}_i \rangle x_i x_i \quad (3)$$

$$= \frac{1}{2} \left( \sum_{i=1}^n \sum_{j=1}^n \sum_{f=1}^k v_{i,f} v_{j,f} x_i x_j - \sum_{i=1}^n \sum_{f=1}^k v_{i,f} v_{i,f} x_i x_i \right) \quad (4)$$

$$= \frac{1}{2} \sum_{f=1}^k \left( \left( \sum_{i=1}^n v_{i,f} x_i \right) \left( \sum_{j=1}^n v_{j,f} x_j \right) - \sum_{i=1}^n v_{i,f}^2 x_i^2 \right) \quad (5)$$

$$= \frac{1}{2} \sum_{f=1}^k \left( \left( \sum_{i=1}^n v_{i,f} x_i \right)^2 - \sum_{i=1}^n v_{i,f}^2 x_i^2 \right) \quad (6)$$

稍微解释一下：

- 从式子.2 到式子.3 的变化，需要注意  $j$  的索引从  $i + 1$  变成了 1，相当于计算了  $\langle \mathbf{v}_i, \mathbf{v}_j \rangle, \langle \mathbf{v}_i, \mathbf{v}_j \rangle$ ，即计算了两遍，但是对角线上的元素只算了一遍，所以还要减去对角线上的元素
- 从式子.3 到式子.4 的变化只是展开了  $\mathbf{v}_i$  而已
- 从式子.4 到式子.6 的变化则是把  $f$  那一层的求和提出来了，再进行了一些变化：

$$\begin{aligned} & \sum_i^n \sum_j^n n v_{if} v_{jf} x_i x_j \\ &= \sum_i^n v_{if} x_i \sum_j^n v_{jf} x_j \\ &= A \cdot \sum_i^n v_{if} x_i \quad (\text{let } A = \sum_j^n v_{jf} x_j) \\ &= A \cdot B \quad (\text{let } B = \sum_i^n v_{if} x_i) \\ &= \left( \sum_i^n v_{if} x_i \right)^2 \end{aligned}$$

这个变化可以看出，FM 的时间复杂度是  $O(kn)$ ，通过梯度下降更新参数时：

$$\frac{\partial}{\partial \theta} \hat{y}(\mathbf{x}) = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \\ x_i, & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^n v_{jf} x_j - v_{if} x_i^2, & \text{if } \theta \text{ is } v_{if} \end{cases}$$

可以看到更新  $v_{if}$  时，需要  $x_i$  不等于 0 的样本。

文中还讨论了 FM 与 SVM 和基于分解的模型的联系与区别。当数据很稀疏时，难以准确的学习到 SVM 的参数、通过变化输入特征，FM 可以与一些基于分解的模型等价。

## 总结

- 将交叉特征的参数矩阵分解，得到每个特征的隐向量，用特征隐向量的内积作为交叉特征的权重
- **怎么保证  $W$  是正定的呢？**
- 分析了 FM 与 SVM 和基于分解的模型的联系

## 29.2 Wide & Deep Learning for Recommender Systems

论文地址：<https://arxiv.org/pdf/1606.07792v1.pdf>

来源：DLRS, 2016

作者：Heng-Tze Cheng, Levent Koc, et al.

源码：[Wide&Deep](#)

关键词：Recommender systems

写于：2021-08-15

该论文 [8] 提出了一个新的模型用于 App 推荐的模型 — Wide&Deep，该模型结合了 Wide 模型 (LR 模型) 和 Deep 模型 (Deep Neural Network)。

**问题定义** 在推荐系统中有一个挑战 — **Memorization & Generalization**。Memorization：学习频繁共现的物品/特征在历史数据中的关系。Generalization：根据关系的传递性，探索新的特征组合，即使未在训练数据中出现或很少出现的特征组合也能给出合适的推荐。Memorization 可以通过 Wide 模型来解决，Generalization 可以通过 Deep 模型来解决。

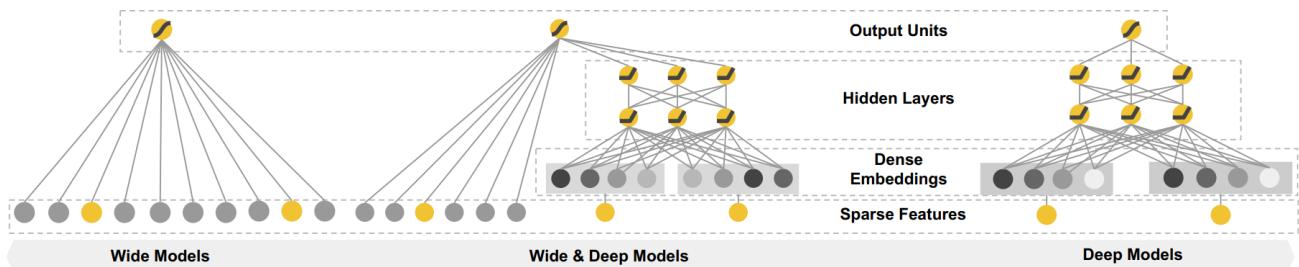


Figure 50: Wide&Deep

## Wide&Deep

**Wide Component** Wide 模型一般是一个广义的线性模型，如 LR 或者包含特征交叉的线性模型。

**Deep Component** Deep 模型是一个基于神经网络的模型。在输入层，Deep 模型将类别变量进行嵌入得到 dense embedding (即 Fig.50 中的 Dense Embedding)。

**Joint of Wide and Deep** Wide 模型和 Deep 模型是联合训练的，Wide&Deep 模型的输出等于 Wide 和 Deep 模型的输出加权求和。论文中使用 FTRL (Follow-the-regularized-leader，其在处理诸如逻辑回归之类的带非光滑正则化项，如 L1 范数的优化问题上性能非常出色) [36] 来优化 Wide&Deep 模型。

在论文中，作者将类被特征转化为 ID，将连续值特征按照值的分位数转换到 [0,1]。

## 总结

- Memorization 与 Generalization 的结合
- Wide&Deep 用于 Ranking

### 29.3 Deep Neural Networks for YouTube Recommendations

论文地址：<https://dl.acm.org/doi/pdf/10.1145/2959100.2959190>

来源：RecSys, 2016

作者：Paul Covington, Jay Adams, Emre Sargin

关键词：recommender system, deep learning, scalability

写于：2021-08-18

该论文 [10] 介绍了 Youtube 进行视频推荐的方法。Youtube 进行视频推荐时分为两个阶段，各自由一个深度模型负责：deep candidate generation model、deep ranking model。

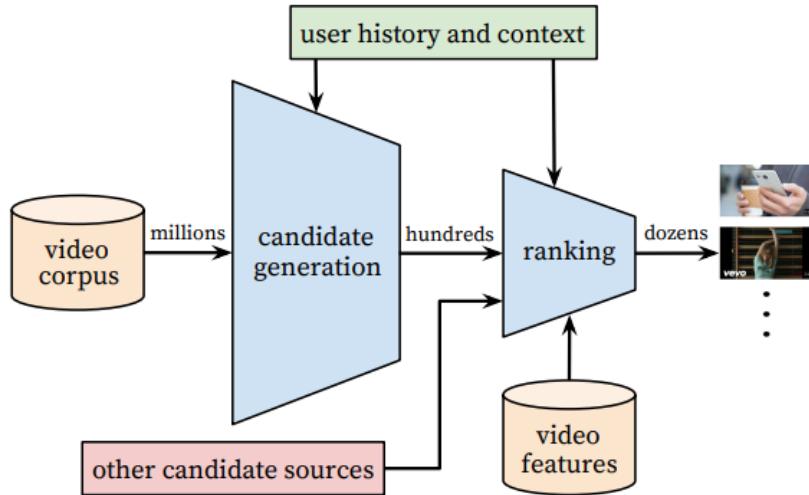


Figure 51: 推荐系统整体架构

**问题定义** 在 Youtube 的视频推荐中主要有三大挑战：

- Scale
- Freshness：每秒都有大量的视频上传到 Youtube 和大量用户产生观看记录，推荐的视频要能够反映最新上传的视频和用户最新的交互行为
- Noise：用户的历史行为是稀疏的且受到很多不可观测的因素的影响，一般难以得到用户的显示的对视频评价数据。需要在这样的数据上构建推荐算法

推荐系统的整体架构如 Fig.51 所示。

**CANDIDATE GENERATION** 这一部分完成相关视频召回，召回阶段要保证高的 Precision (尽量把相关的挑出来)。文中将召回作为一个多分类任务：

$$P(w_t = i | U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$

$w_t$  表示在时刻  $t$  看的视频的类别，每个视频  $i$  就是一个类别（没错，这是一个类别多达百万的多分类任务）， $V, U, C$  分别表示视频集、用户、上下文， $u \in \mathbb{R}^n, v_i \in \mathbb{R}^n$  分别表示  $<$  用户，上下文  $>$  的向量、视频的向量。这一步模型的结构如 Fig.52 所示，在最后一层输出的是 *user vector u*，这个模型的目标是：**学习一个函数，该函数以用户历史行为和上下文为输入，输出用户的 embedding**

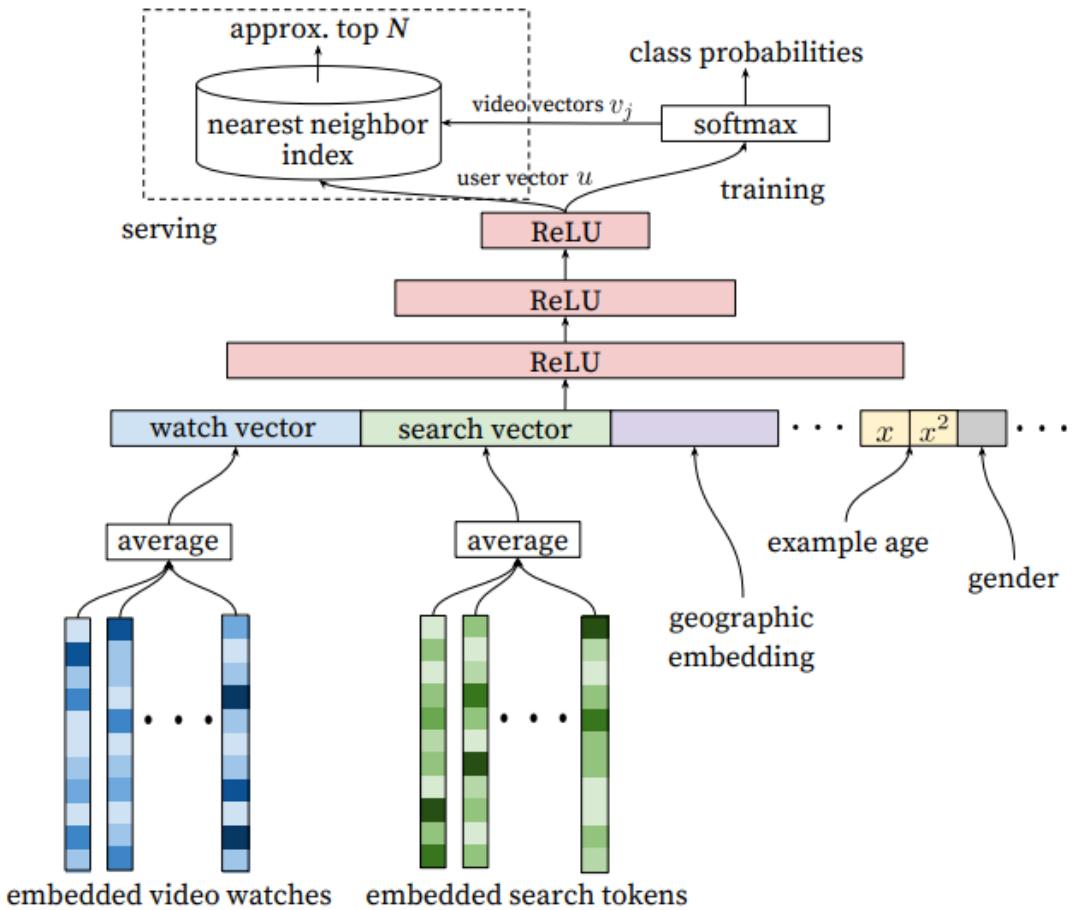


Figure 52: Deep candidate generation model architecture

嗯？不是说好的多分类吗，怎么变成了用户 embedding 了？在 Fig.52 的顶部，最后一层的 *Relu* 的输出分别给到了 serving 和 training 两部分。送往 training 的部分确实会根据用户 embedding 计算类别概率，但是在 serving 时，是召回 top N 个视频，这时候只需要把用户 embedding 作为 query 在视频集里找到 top N 个相似的视频即可。

**训练数据的生成** 为每个用户生成相同数量的训练样本，避免过于活跃的用户对损失函数产生过大的影响。而且由于涉及到极大的多分类问题，因此涉及到 softmax 的计算，为了加快速度，采用了负采样。

**特殊的特征** 每秒都有大量的视频上传到 Youtube，推荐最新上传的视频给用户是很重要的。一个视频在 Youtube 上的热门程度是不平稳的，但是召回模型对一个视频的预测的概率（热门程度）是平稳的，为了矫正这个问题，增加了一个 *exampleage* 特征，在训练时，样本的 *exampleage* 特征取值为  $t_{max} - t_N$ ，其中  $t_{max}$  表示训练样本中最大的视频上传时间， $t_N$  是样本的上传时间；在 serving 时，该特征为 0.

**RANKING** 这一部分对召回的视频进行排序，排序的依据是视频的期望观看时长，排序时要尽量保证高的 Recall。如 Fig.53 所示，Ranking 模型通过 weighted logistic (在 weighted LR 中，输出样本  $i$  的 odds 会变成原来的  $w_i$  倍， $w_i$  为  $i$  的权重 ) 进行训练。

Ranking 阶段的 weighted LR 中将正样本的观看时长作为样本权重，负样本的权重为 1。通过训练得到了 weighted LR 的参数，得到每个视频被点击的概率，由于权重是视频时长，则 odds 就是该视频期望播放时长。

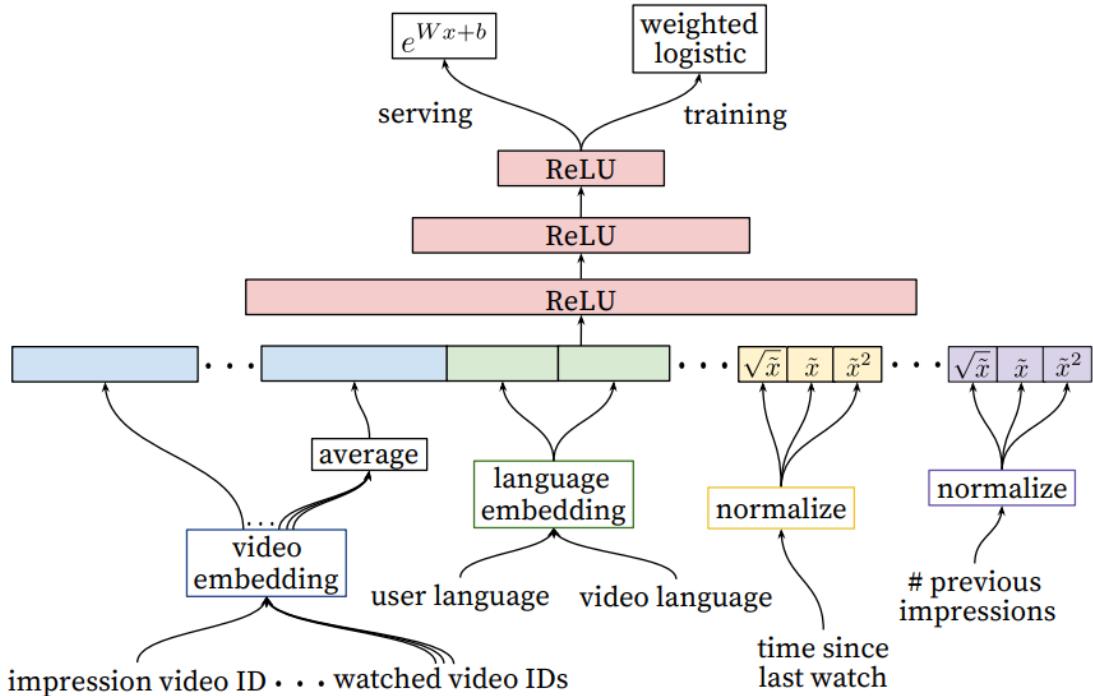


Figure 53: Deep ranking network architecture

**特征表示** 处理特征时忽略了时序，这样或许有助于模型的泛化。为一个视频打分时，该用户与这个视频（或相似视频）之前的交互数据最具参考价值。

## 总结

- 将召回阶段作为一个极多类别的分类任务
- 排序阶段预测每个视频的期望播放时长
- 特征工程功不可没

## 29.4 Field-aware Factorization Machines for CTR Prediction

论文地址：<https://www.csie.ntu.edu.tw/~cjlin/papers/ffm.pdf>

来源：RecSys, 2016

作者：Yuchin Juan, Yong Zhuang, et al.

源码：[libffm](#)

关键词：Machine learning, Click-through rate prediction, Computational advertising, Factorization machines

写于：2021-08-28

该论文 [22] 提出了 FM 模型的一个变种 — FFM (Field-aware Factorization Machines)。FFM 最初是应用的 CTR 预测任务上，模型与 FM 很类似。FFM 也做了二阶特征交叉，交叉特征的权重的计算方式与 FM 略微不同。

**问题定义** 解决 CTR 预测问题中特征稀疏、特征交叉的问题。

**FFM** FFM 中将  $n$  个特征进行了分组，得到  $f$  个 field，每个特征在每个 field 中都有一个  $k$  维的向量表示。对于交叉特征  $x_{j_1} \cdot x_{j_2}$ ，其中  $x_{j_1}$  属于 field  $f_1$ ， $x_{j_2}$  同理，其权重计算方式为： $\mathbf{w}_{j_1, f_2} \cdot \mathbf{w}_{j_2, f_1}$ ，其中  $\mathbf{w}_{j_1, f_2} \in \mathbb{R}^n$  表示特征  $x_{j_1}$  在 field  $f_2$  中的向量表示。

---

### Algorithm 1 Training FFM using SG

---

```

1: Let  $G \in R^{n \times f \times k}$  be a tensor of all ones
2: Run the following loop for  $t$  epochs
3: for  $i \in \{1, \dots, m\}$  do
4:   Sample a data point  $(y, \mathbf{x})$ 
5:   calculate  $\kappa$ 
6:   for  $j_1 \in$  non-zero terms in  $\{1, \dots, n\}$  do
7:     for  $j_2 \in$  non-zero terms in  $\{j_1 + 1, \dots, n\}$  do
8:       calculate sub-gradient by (5) and (6)
9:       for  $d \in \{1, \dots, k\}$  do
10:      Update the gradient sum by (7) and (8)
11:      Update model by (9) and (10)

```

---

Figure 54: Training FFM with SGD

使用 SGD 训练 FFM 的过程如 Fig.54 所示，其中引用的公式 (5), (6), (7), (8), (9), (10) 分别是：

$$\begin{aligned}
\mathbf{g}_{j_1, f_2} &\equiv \nabla_{\mathbf{w}_{j_1, f_2}} f(\mathbf{w}) = \lambda \cdot \mathbf{w}_{j_1, f_2} + \kappa \cdot \mathbf{w}_{j_2, f_1} x_{j_1} x_{j_2} \\
\mathbf{g}_{j_2, f_1} &\equiv \nabla_{\mathbf{w}_{j_2, f_1}} f(\mathbf{w}) = \lambda \cdot \mathbf{w}_{j_2, f_1} + \kappa \cdot \mathbf{w}_{j_1, f_2} x_{j_1} x_{j_2} \\
(G_{j_1, f_2})_d &\leftarrow (G_{j_1, f_2})_d + (g_{j_1, f_2})_d^2 \\
(G_{j_2, f_1})_d &\leftarrow (G_{j_2, f_1})_d + (g_{j_2, f_1})_d^2 \\
(w_{j_1, f_2})_d &\leftarrow (w_{j_1, f_2})_d - \frac{\eta}{\sqrt{(G_{j_1, f_2})_d}} (g_{j_1, f_2})_d \\
(w_{j_2, f_1})_d &\leftarrow (w_{j_2, f_1})_d - \frac{\eta}{\sqrt{(G_{j_2, f_1})_d}} (g_{j_2, f_1})_d
\end{aligned}$$

Fig.54 中的 (5) 计算的是损失函数（包含了 l2 正则）关于  $\mathbf{w}_{j_1, f_2}$  的梯度，(6) 同理。(7) 则计算的是 (5) 中梯度的各个维度的平方和， $G$  初始化为 1，避免计算完后  $G$  太小导致  $G^{-\frac{1}{2}}$  太大，(8) 同理。(9) 则是对  $\mathbf{w}_{j_1, f_2}$  逐维更新，其中  $\eta$  自定义的学习率，这就是实际更新交叉特征参数的步骤，(7) 中计算的  $G$  相当于对梯度向量归一化（个人觉得论文上公式是不是写错了，(7), (8) 中应该是  $G_{j_1, f_2}$  而不是  $(G_{j_1, f_2})_d$ ）。

**为什么要 field-aware ?** FM 相当于 FFM 的一个特例，只有 1 个 field。与 FM 相比，FFM 将特征分成多个 fields，每个特征在每个 field 中都有一个向量表示（从这个角度看有点像解耦表征）。当一个特征  $x_{j_1}$  与其他特

征组合时，FM 只有一个向量用于计算，而 FFM 中，会根据另一个特征所属的 field 使用不同的向量进行计算，相当于对模型表达能力的一个细化，增强模型的表达能力。当然，FFM 的参数量增加了，而且不可以像 FM 那样化简，因此其时间复杂度为  $O(\bar{n}^2k)$ ，但 FFM 中隐向量的维度要远小于 FM 的维度。

**Assign field to feature** 既然每个特征都属于一个 field，那怎么划分呢？对于类别特征，很显然，一般都属于同一个 field。对于数值特征，可以对其进行离散化后作为类别特征处理。还有一种就是单一特征的样本（只有一个特征），例如特征是文本的情况下，每个文本基本都是不一样的，如果都划分到一个 field 则退化成了 FM，如果每个文本都是一个 field 则是很不实际的，因为一般样本的数量是很大的，对于这个问题论文中并没有进行给出方法，个人认为：将文本处理成向量，可以是 bag of words，一些统计量、embedding 等。

## 总结

- 将特征划分到 fields，增强模型的表达能力，一个特征与不同 field 的特征交叉时使用不同的向量表示
- 能够应对较稀疏的数据，当数据不是很稀疏时可能提升不是很明显
- 数值特征和单一特征划分 field 是个值得探讨的问题

## 29.5 DeepFM: A Factorization-Machine based Neural Network for CTR Prediction

论文地址：<https://www.ijcai.org/Proceedings/2017/0239.pdf>

来源：IJCAI, 2017

作者：Huifeng Guo, Ruiming Tang, et al.

关键词：CTR, FMs, Wide&Deep

写于：2021-08-31

该论文 [20] 提出了基于 FM 的深度模型 — DeepFM，用于解决 CTR 问题。DeepFM 与 Wide&Deep 模型很像，但也有一些不同。DeepFM 将 Wide&Deep 中的 LR 替换为 FM；不需要做特征工作；Wide 和 Deep 部分的输入是共享的。

**问题定义** 针对 CTR 问题，样本的特征可以分成多个 field， $x = [x_{field_1}, x_{field_2}, \dots, x_{field_m}]$ ， $x_{field_i}$  是  $field_i$  的向量表示，如 one-hot 编码。

**DeepFM** DeepFM 的整体结构如 Fig.55 所示。DeepFM 由两个部分组成：FM component 和 Deep component，这两部分共享输入。自底向上模型包含了 Sparse Feature、Dense Embedding、FM Layer、Hidden Layer 和 Output Units。

对于 Sparse Feature 中的每一个特征，有一个权重与之相乘得到线性部分，也就是一阶特征；特征隐向量用于计算交叉特征之间的重要性，也就是二阶特征。特征隐向量也会作为 Deep 部分的输入。FM 的输出与 Deep 的输出相加后经过 Sigmoid 函数后得到最终的输出。

$$\hat{y} = \text{sigmoid}(y_{FM} + y_{DNN})$$

**FM Component** FM 部分的结构如 Fig.56 所示，与常规的 FM 没有很大区别。**有个问题，从 Fig.56 看，各个 field 的 Dense Embedding 进行内积作为特征的权重，这么说的话每个 field 一个隐向量，Dense Embedded 就是这个隐向量？**

在 FM 部分，可以学习到一阶和二阶的特征。

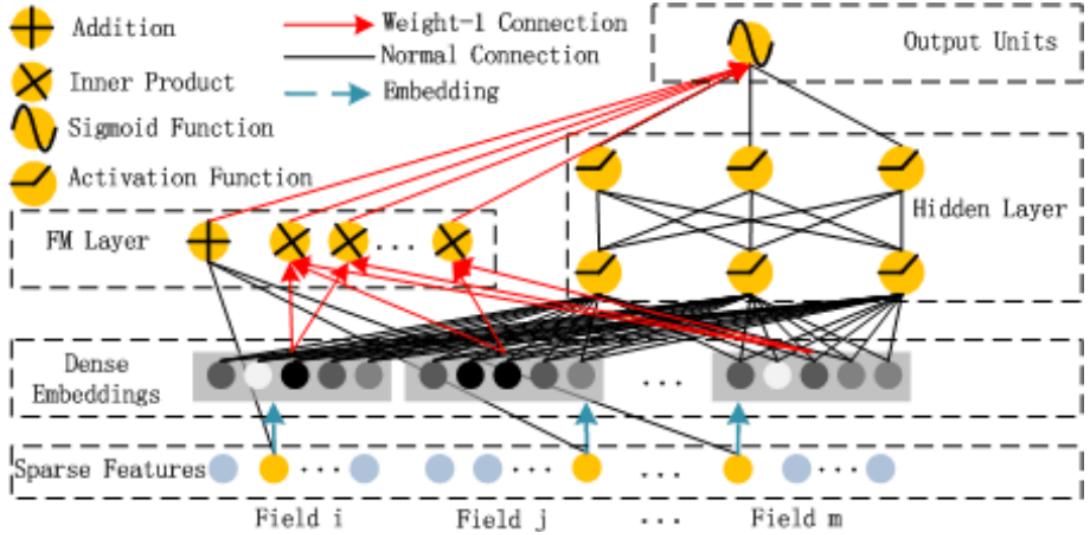


Figure 55: Wide & deep architecture of DeepFM

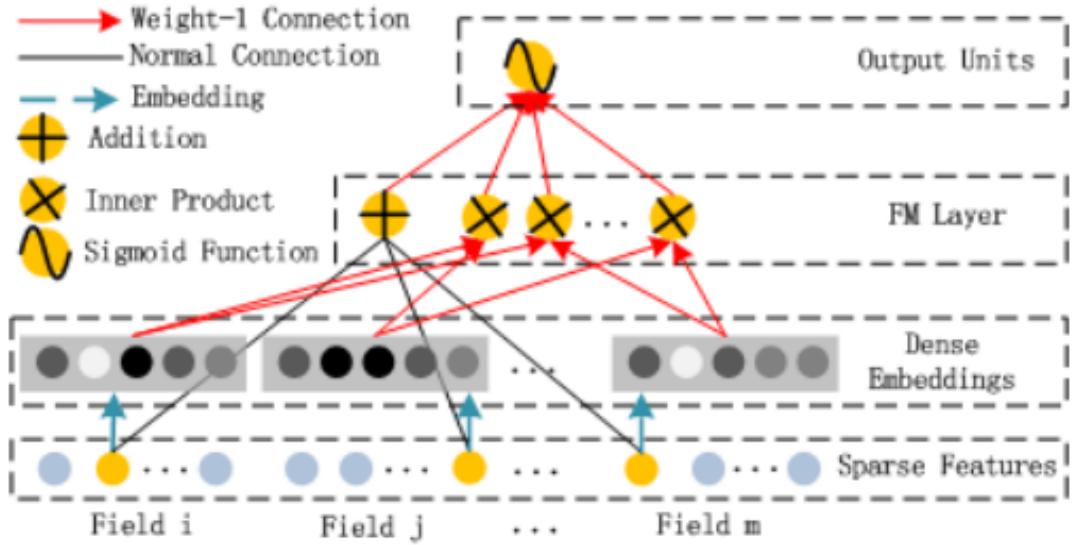


Figure 56: The architecture of FM

**Deep Component** Deep 部分的结构如 Fig.57 所示。由于大部分特征都是稀疏的，为了输入神经网络中，Dense Embedding 将稀疏的特征转化成稠密的向量。Dense Embedding 的结构如 Fig.58 所示。

Dense Embedding 层将  $x_{field}$  向量编码成低维的稠密向量，其计算过程如 Fig.59 所示。 $e_i = x_i \cdot \mathbf{V}$ ，其中  $x_i \in \mathbb{R}^n$  是  $field_i$  的向量表示， $\mathbf{V} \in \mathbb{R}^{n \times k}$  每行表示一个特征的隐向量， $n$  表示  $field_i$  的特征数。很显然，Dense Embedding 层将  $\mathbf{V}$  作为参数。如果  $x_i$  是 one-hot，则  $e_i$  表示第  $i$  个特征的隐向量，即将不为零的那个特征的隐向量作为其 Dense Embedding 的输出。最终 Dense Embedding 的输出为： $a^{(0)} = [e_1, e_2, \dots, e_m]$ ， $a^{(0)}$  作为 Hidden Layer 的输入，接下来就是常规操作啦。

在 Deep 部分，可以学习到高阶的交叉特征。

最终，FM 部分和 Deep 部分是联合训练的，整个模型的参数包括  $w_i, \mathbf{V}, (\mathbf{W}^{(l)}, b^{(l)})$ 。

## 总结

- 与现有的一些模型，如 FNN，不需要预先训练 FM 部分的隐向量，DeepFM 通过联合训练得到隐向量
- DeepFM 的 FM 和 Deep 部分共享输入

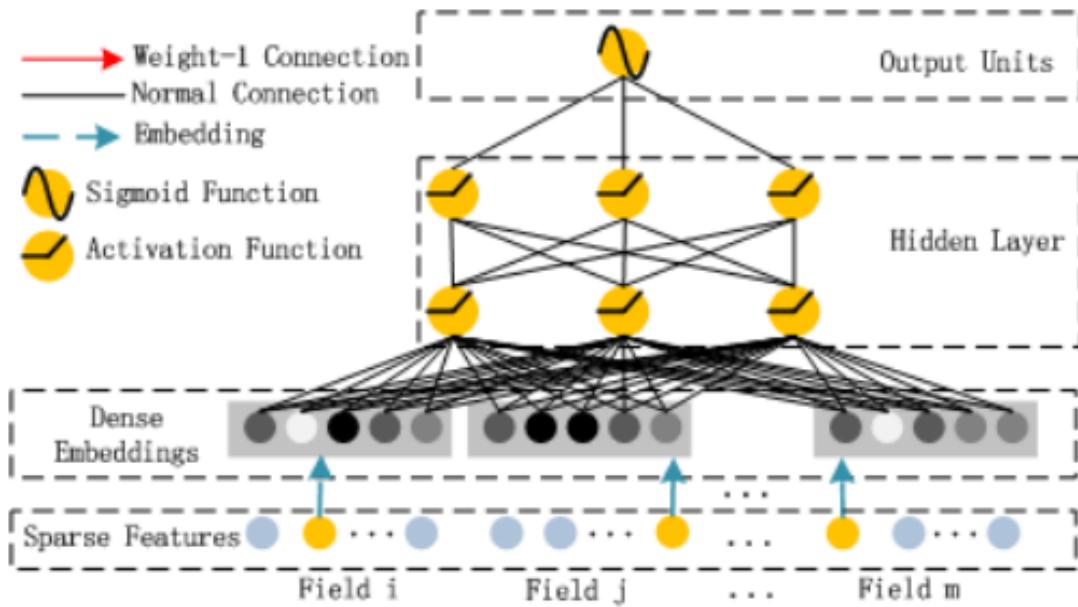


Figure 57: The architecture of DNN

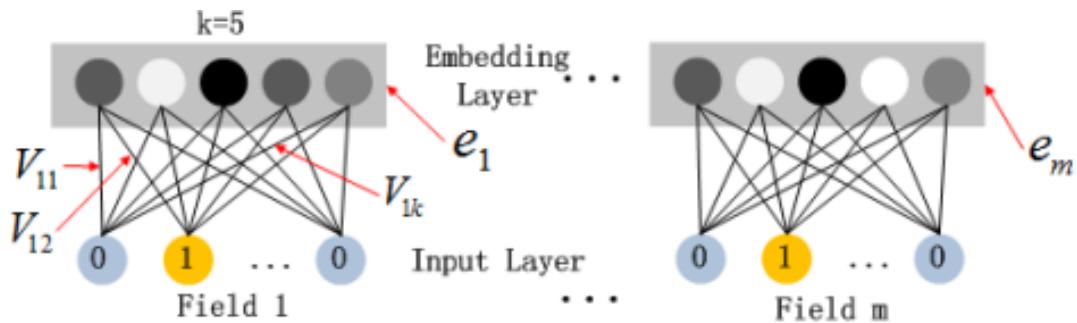


Figure 58: The structure of the embedding layer

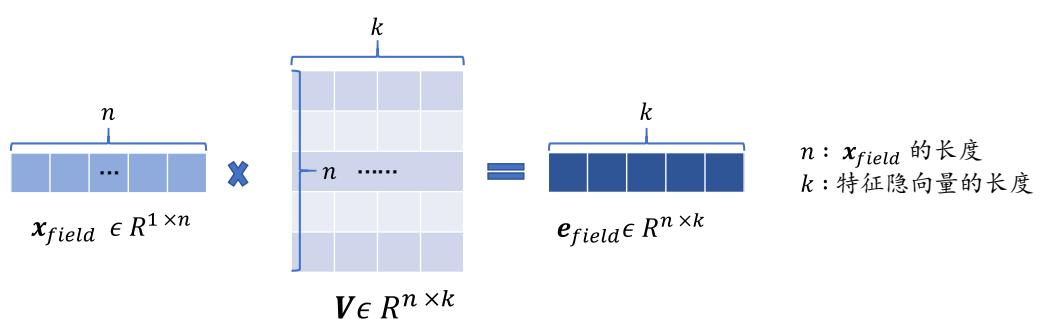


Figure 59: Dense Embedding 层计算方式

## 30 简读论文

### 30.1 Embedding Words in Non-Vector Space with Unsupervised Graph Learning

论文地址: <https://arxiv.org/abs/2010.02598>

源码地址: [graph-glove](#)

关键词: **embedding, representation learning**

写于: 2020-10-08

传统的 word embedding 通常是将词嵌入到向量空间中, [39] 指出 words 可以形成具有隐式层次结构的图, 词向量的质量与选取什么样的向量空间有很大的关系。针对这个问题, 该论文 [48] 提出了 GraphGlove, 将词嵌入到带权图 (weighted graph) 中。在 word 相似性及相关任务中, 该论文提出的方法取得了比嵌入到向量空间中更好的效果。

**方法—GraphGlove** 每个 word 视作带权图 (weighted graph) 中的一个结点, **词之间的距离使用图中结点之间的 shortest path 来表示 (与欧氏空间中距离的定义不同)**。使用 PRODIGE[35] 从数据中学习, 可以得到一个带权图以及图中边的权重; 再将学习得到的带权图用于 GloVe[44] 算法, 学习得到在非欧空间中的词向量, 关键之处是替换 GloVe 的损失函数中词之间距离为图中结点之间的最短距离。

该论文与其他 embedding 方法不同的之处: 将 word 嵌入到非欧空间, 学习到的带权图就是这个非欧空间。

### 30.2 Accurate, Efficient and Scalable Training of Graph Neural Networks

论文地址: <https://arxiv.org/abs/2010.03166>

源码地址: [GraphSAINT](#)

关键词: **GRL, GNN, Graph sampling, Graph partitioning, Memory optimization**

写于: 2020-10-08

该论文 [70] 针对 GNN 的训练方法提出了一些改进算法, 主要的改进点是: 通过采样子图及关键步骤 (如汇聚结点的特征) 并行化。论文中针对多种 GNN 模型设计了多种并行策略。

### 30.3 Learning to Represent Image and Text with Denotation Graph

论文地址: <https://arxiv.org/abs/2010.02949>

源码地址: [DG](#)

关键词: **embedding, multimodal, Denotation Graph**

写于: 2020-10-08

该论文 [71] 提出了一种表征多模态 (图像 + 文本) 数据的方法。存在着大量对齐的“视觉 + 文本”的数据, 如包含描述的图片、视频、带字幕的电影等。学习如何表示这种视觉和文本存在明显的语义联系的数据是有很大引用价值的, 如通过文本搜索图片、视频, 为视频/图片添加描述, 通过语言查询视频/图片中的信息, 可视化的问答系统等。

**方法** 论文中使用 denotation graph(DG)[69] 来表示“图像 + 文本”的数据。该论文中的 DG 中每个结点包含一个文本描述和一系列与之对应的图像, 图中的边是有向边, 从语义上抽象的结点指向更具体地结点。如 Fig.60 所示, DG 中抽象的结点具有更一般的概念 (如图中的 person, instrument), 抽象结点会指向更具体地结点 (如 play percussion instrument), 每个结点会包含结点概念所对应的一些列图片。为什么要将文本与图像对应起来

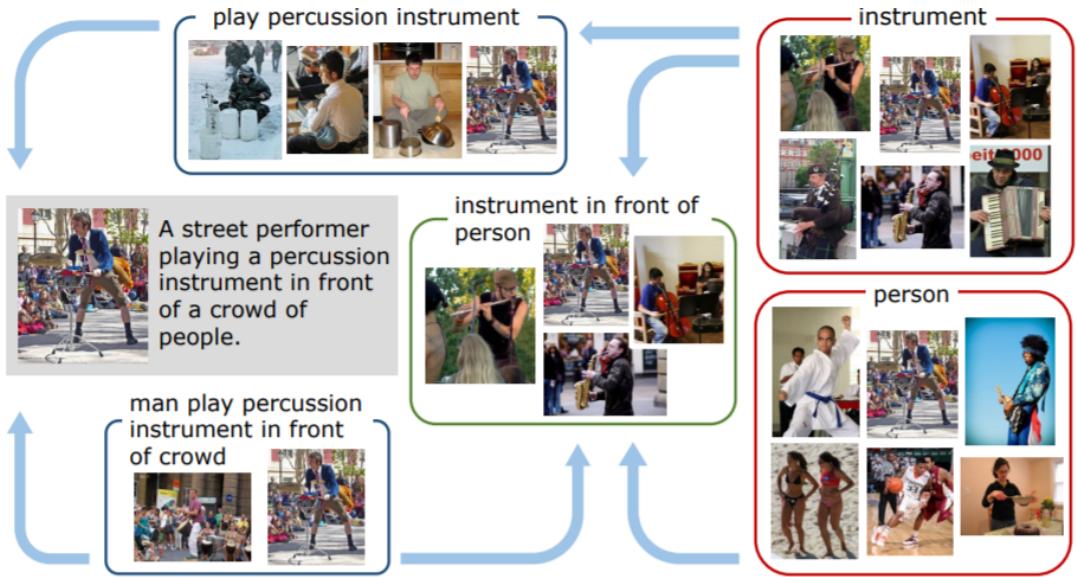


Figure 60: denotation graph extracted from the FLICKR30K dataset

呢？论文中基于这样的一个假设：图片与文本对应的一致性有利于下游任务。在得到上述的 DG（构建 DG 可参考[这里](#)和 [69]）后，在 DG 基础上学习图片和文本地表征。

### 30.4 Towards Expressive Graph Representation

论文地址：<https://arxiv.org/abs/2010.05427>

源码地址：[ExpGNN](#)

关键词：GNN

写于：2020-10-13

该论文 [34] 针对 GNN 存在的 non-injective 问题进行了改进，在理论上进行分析并设计了连续的、injectiv 的邻居信息聚集函数，尽可能将不同的结点映射到不同的表征，相似的结点表征映射到相近的表征。聚集函数的连续性保证输入的微小变化能够导致表征的变化。论文中对 GNN 中的邻居结点信息聚集函数、结点自身信息与聚集后的邻居信息结合函数、根据结点表征生成图表征的函数进行了设计，使其 sh 是 injective 且连续的。特别的针对邻居信息聚集函数，设计了两个版本，一个是固定的聚集函数，一个是被参数化的 MLP。

### 30.5 Enhancing Extractive Text Summarization with Topic-Aware Graph Neural Networks

论文地址：<https://arxiv.org/abs/2010.06253>

关键词：Document Summarization, GAT, BERT

写于：2020-10-15

该论文 [11] 针对抽取式文本摘要问题提出了一个基于 GNN 的方法。现有的抽取式的文档摘要算法中，是通过从文档中抽取一个句子序列作为文档的摘要，但是通常没有考虑句子之间的关系，论文针对这个问题，提出了 Topic-GraphSum 模型。

如 Fig.61 所示，Topic-GraphSum 模型主要分为三个部分，NTM (Neural Topic Model) 用于从文档中抽取话题，Document Encoder 用于生成每个句子的一个表征，再用前两部生成的 topics 和句子构造一个 Heterogeneous

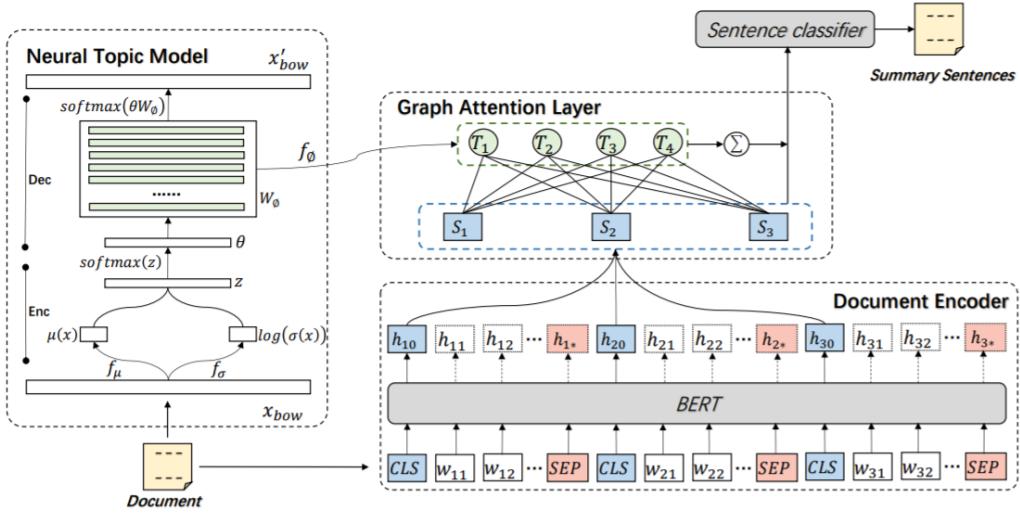


Figure 61: Overview of Topic-GraphSum

Graph, 使用 GAT 对该异构图进行学习, 再将每个句子的表征输入到 Sentence Classifier 中, 输出每个句子是否在 Summarization 中。

### 30.6 Editing Graphs to Satisfy Diversity Requirements

关键词: Additional Edges, Graph Editing Problem, Graph Edit Operations, General Factor Problem, Allowable Edit

写于: 2020-10-20

论文大意: 图中每个结点, 其邻居数量处于某个区间内, 通过边的增加/删除来将一个图转化为另一个图。

### 30.7 On the exact computation of the graph edit distance

关键词: Graph edit distance, Exact algorithms, Integer programming

写于: 2020-10-20

论文对几个主流的 GED 精确/近似算法进行了总结, 并在这些算法的基础上进行了一定的改进。但依然是一个精确的 GED 求解方法。并证明了当图的结点数超过 16 时, 算法将难以运行下去。

### 30.8 Retrieval-Augmented Generation for Code Summarization via Hybrid GNN

论文地址: <https://openreview.net/pdf?id=zv-typ1gPxA>

来源: ICLR, 2021

作者: Shangqing Liu, et al.

关键词: GNN, Code Summarization

写于: 2021-07-19

该论文 [30] 的目标：给定某种语言编写的函数代码，生成这个函数的自然语言描述。自动的代码摘要主要有两种方式：1) Retrieval-based (抽取式)；2) Generation-based (生成式)。该论文提出了一种结合这两种方法的增强的抽取式方机制来进行代码总结。

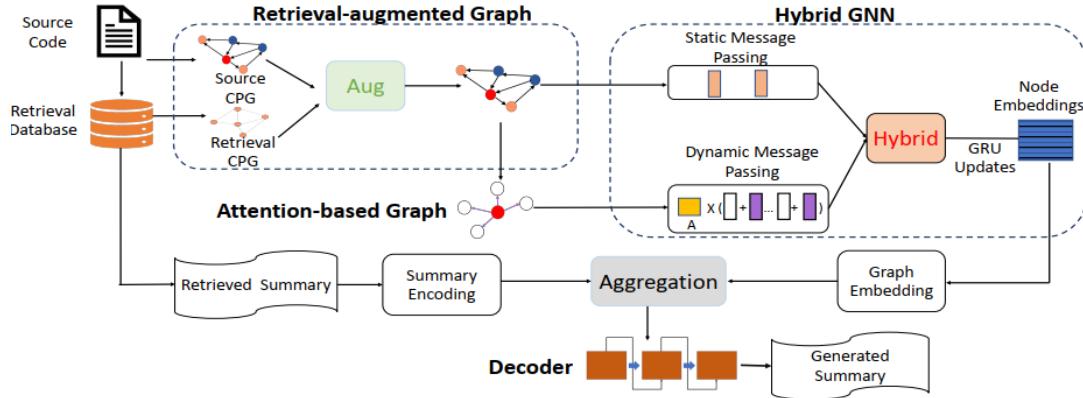


Figure 62: Overview of HGNN

论文提出的方法是 Hybrid GNN (Fig.30.8)，首先根据代码的抽象语法树表示成 Graph (Fig.30.8)，再从代码数据集中找到与当前代码最相似的代码（除去当前的函数）及对应的 summarization，当前代码的 Graph 经过最相似的代码的 Graph 增强后得到 static graph，经过注意力机制后生成 dynamic graph，在 static/dynamic graph 上进行 message passing，再通过 Hybrid massage passing (即将同一个结点在 static 和 dynamic graph 中的表征融合后再进行 message passing)。除此之外，还会利用找到的最相似的代码的 summarization 的表征，将最终得到的结点表征与 summarization 表征进行拼接后输入基于注意力的 LSTM，得到目标代码的表征。

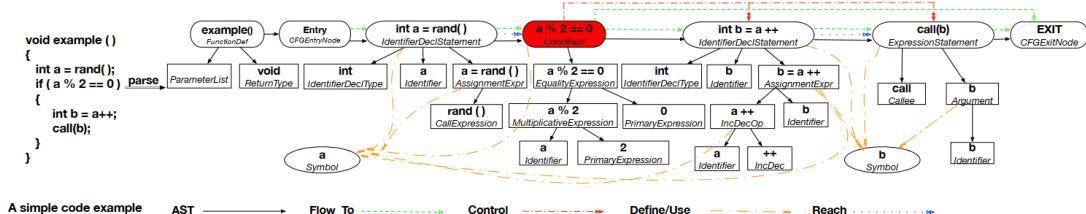


Figure 63: An Example of Code Property Graph(CPG)

## References

- [1] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.
- [2] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net, 2019.
- [3] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn - a neural network approach to fast graph similarity computation. *WSDM*, pages 384–392, 2019.
- [4] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. Unsupervised inductive graph-level representation learning via graph-graph proximity, 2019.

- [5] Yunsheng Bai, Hao Ding, Yizhou Sun, and Wei Wang. Convolutional set matching for graph similarity, 2018.
- [6] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
- [7] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. *ICLR*, 2018.
- [8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide deep learning for recommender systems. *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 2016.
- [9] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.
- [10] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. *RecSys*, pages 191–198, 2016.
- [11] Peng Cui, Le Hu, and Yuanchao Liu. Enhancing extractive text summarization with topic-aware graph neural networks, 2020.
- [12] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations (ICLR)*, 2020.
- [13] S. Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, pages 1944–1957, 2007.
- [14] Carl Doersch. Tutorial on variational autoencoders, 2016.
- [15] Locatello Francesco, Poole Ben, Rätsch Gunnar, Schölkopf Bernhard, Bachem Olivier, and Tschannen Michael. Weakly-supervised disentanglement without compromises. *ICML*, pages 6348–6359, 2020.
- [16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680, 2014.
- [17] kaiming he, xiangyu zhang, shaoqing ren, and jian sun. Deep residual learning for image recognition. *CVPR*, 2016.
- [18] Hengguan Huang, Fuzhao Xue, Hao Wang, and Ye Wang. Deep graph random process for relational-thinking-based speech recognition. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4531–4541. PMLR, 2020.
- [19] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and R. Austin Benson. Combining label propagation and simple models out-performs graph neural networks. *international conference on learning representations*, 2020.

- [20] Guo Huifeng, Tang Ruiming, Ye Yunming, Li Zhenguo, and He Xiuqiang. Deepfm: A factorization-machine based neural network for ctr prediction. *IJCAI International Joint Conference on Artificial Intelligence*, pages 1725–1731, 2017.
- [21] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017.
- [22] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. *RecSys*, pages 43–50, 2016.
- [23] Hosein Amir Khasahmadi, Kaveh Hassani, Parsa Moradi, Leo Lee, and Quaid Morris. Memory-based graph networks. *ICLR*, 2020.
- [24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR 2014 : International Conference on Learning Representations (ICLR) 2014*, 2014.
- [25] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.
- [26] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.
- [27] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.
- [28] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.
- [29] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. In *ICLR 2018 : International Conference on Learning Representations 2018*, 2018.
- [30] Shangqing Liu, Yu Chen, Xiaofei Xie, Kai Jing Siow, and Yang Liu. Retrieval-augmented generation for code summarization via hybrid gnn. *ICLR*, 2021.
- [31] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *computer vision and pattern recognition*, pages 3431–3440, 2015.
- [32] Guixiang Ma, Nesreen K. Ahmed, Theodore L. Willke, and Philip S. Yu. Deep graph similarity learning: A survey, 2020.
- [33] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders, 2016.
- [34] Chengsheng Mao, Liang Yao, and Yuan Luo. Towards expressive graph representation, 2020.
- [35] Denis Mazur, Vage Egiazarian, Stanislav Morozov, and Artem Babenko. Beyond vector spaces: Compact data representation as differentiable weighted graphs, 2019.
- [36] Brendan H. McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. *AISTATS*, pages 525–533, 2011.
- [37] Diego Mesquita, Amauri Souza, and Samuel Kaski. Rethinking pooling in graph neural networks. *NeurIPS*, 2020.
- [38] Silvio Micali and Allen Zeyuan Zhu. Reconstructing markov processes from independent and anonymous experiments. *Discrete Applied Mathematics*, pages 108–122, 2016.

- [39] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.
- [40] Jovana Mitrovic, Brian McWilliams, Jacob C Walker, Lars Holger Buesing, and Charles Blundell. Representation learning via invariant causal mechanisms. In *International Conference on Learning Representations*, 2021.
- [41] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *CoRR*, abs/1707.05005, 2017.
- [42] Peter O’Hearn, John Reynolds, and Hongseok Yang. Local reasoning about programs that alter data structures. In Laurent Fribourg, editor, *Computer Science Logic*, pages 1–19, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- [43] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, and E. Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. *arXiv: Learning*, 2019.
- [44] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [45] Steffen Rendle. Factorization machines. *ICDM*, pages 995–1000, 2010.
- [46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science*, pages 234–241, 2015.
- [47] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. 2020.
- [48] Max Ryabinin, Sergei Popov, Liudmila Prokhorenkova, and Elena Voita. Embedding words in non-vector space with unsupervised graph learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
- [49] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 519–527, 2020.
- [50] Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. *international conference on learning representations*, 2021.
- [51] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011.
- [52] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.
- [53] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *27th International Conference on Artificial Neural Networks (ICANN)*, pages 412–422, 2018.

- [54] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. *international conference on learning representations*, 2020.
- [55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, N. Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 30 (NIPS 2017)*, pages 5998–6008, 2017.
- [56] Sheng Wan, Shirui Pan, Jian Yang, and Chen Gong. Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. 2020.
- [57] Lichen Wang, Bo Zong, Qianqian Ma, Wei Cheng, Jingchao Ni, Wenchao Yu, Yanchi Liu, Dongjin Song, Haifeng Chen, and Yun Fu. Inductive and unsupervised representation learning on graph structured objects. In *International Conference on Learning Representations*, 2020.
- [58] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2021.
- [59] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks, 2015.
- [60] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2020.
- [61] Haoyan Xu, Runjian Chen, Yunsheng Bai, Ziheng Duan, Jie Feng, Ke Luo, Yizhou Sun, and Wei Wang. Hierarchical large-scale graph similarity computation via graph coarsening and matching, 2020.
- [62] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.
- [63] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *AAAI*, pages 7444–7452, 2018.
- [64] Pinar Yanardag and V. N. S. Vishwanathan. Deep graph kernels. *ACM Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.
- [65] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling, 2019.
- [66] Jiaxuan You, Jure Leskovec, Kaiming He, and Saining Xie. Graph structure of neural networks, 2020.
- [67] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation, 2019.
- [68] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pages 5694–5703, 2018.
- [69] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

- [70] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Accurate, efficient and scalable training of graph neural networks. *Journal of Parallel and Distributed Computing*, 147:166–183, Jan 2021.
- [71] Bowen Zhang, Hexiang Hu, Vihan Jain, Eugene Ie, and Fei Sha. Learning to represent image and text with denotation graph, 2020.