量的维度。这样的话，图融合可以理解为一个函数$f(\cdot)$，该函数输出一个新的邻接矩阵$A_{fusion} \in R^{N\times N}$，可以理解为加权图$G_{fusion} = (V, E_{fusion})$，即$A_{fusion} = f(A, X)$。

阶段二：频谱粗化（Spectral Coarsening）

基于新兴的图信号处理技术，采用高效的局部频谱嵌入方法识别节点簇。简单来说，就是对k个随机向量应用简单平滑（低通图滤波）函数，得到k维图嵌入的平滑向量，这样可以在线性的时间内实现。

考虑一个由图拉普拉斯矩阵（图拉普拉斯矩阵=度矩阵-邻接矩阵）的特征向量$\boldsymbol{u}$线性组合表示的随机向量$\boldsymbol{x}$（图信号），采用低通图滤波器快速滤除随机图信号的高频分量或图拉普拉斯矩阵的高特征值对应的特征向量，通过在$\boldsymbol{x}$上应用平滑函数，可以得到一个平滑向量$\tilde{\boldsymbol{x}}$，他基本上是前几个特征向量的线性组合：

$$\boldsymbol{x} = \sum_{i=1}^{N} \alpha_i \boldsymbol{u}_i \quad \xrightarrow{smoothing} \quad \tilde{\boldsymbol{x}} = \sum_{i=1}^{n} \tilde{\boldsymbol{\alpha}}_i \boldsymbol{u}_i \ , n \ll N$$

具体来说，使用高斯迭代将线性方程组$L_G x^{(i)} = 0$ 求解到一组t个初始随机向量$T = \left(x^{(1)}, \cdots, x^{(t)}\right)$，这组向量正交于全 1 向量，满足$\mathbf{1}^T x^{(i)} = 0$，其中$L_G$是原图或融合图的拉普拉斯矩阵。

根据T中的平滑向量，作者将每个节点都嵌入 T 维空间，这样如果节点 p 和 q 的低维嵌入向量$X_p \in R^t$和$X_q \in R^t$高度相关，那么可以认为他们在频谱上相似。这里的节点之间的距离采用相邻节点 p 和 q 的频谱节点亲和度$a_{p,q}$来衡量:

$$a_{p,q} = \frac{\left|T_{p,:}, T_{q,:}\right|^2}{\left\|T_{p,:}\right\|^2 \left\|T_{q,:}\right\|^2}, \quad \left(T_{p,:}, T_{q,:}\right) = \sum_{k=1}^{t} \left(x_p^{(k)} \cdot x_q^{(k)}\right)$$

在确定了节点聚合方法后，就可以在两个粗化级别i和$i+1$之间获得图映射算子H_i^{i+1}，其中H_i^{i+1}大小为$\left|V_{G_{i+1}}\right| \times \left|V_{G_i}\right|$。如果$G_i$中的节点$q$聚合到$G_{i+1}$中的节点$p$上，则$(H_i^{i+1})_{p,q} = 1$，否则为 0。利用$H_i^{i+1}$来构造一系列频谱缩减图$G_1, G_2, \cdots, G_l$，较粗的图拉普拉斯矩阵$L_{G_{i+1}}$可以通过公式计算：

$$L_{G_{i+1}} = H_i^{i+1} L_{G_i} H_{i+1}^i, \ H_{i+1}^i = (H_i^{i+1})^T$$

这能高效地保留图的频谱特性，或者说是全局特性

阶段三：图嵌入（Graph Embedding）

在构造了粗化后的图之后，可以通过$E_l = g(G_l)$，其中$g(\cdot)$可以是任意无监督的图嵌入方法。

阶段四：嵌入优化（Embedding Refinement）

对于给定在第$i+1$级的图G_{i+1}的节点嵌入E_{i+1}，使用相应的投影算子H_{i+1}^i将E_{i+1}投影到G_i上，即得到第i级的精细图：

$$\tilde{E}_i = H_{i+1}^i E_{i+1}$$

由于投影算子的性质，在粗图中嵌入的节点会直接复制到上一级细图中同一聚合集的节点中，在这种情况下，精细图中频谱相似的节点如果在粗化阶段聚合成单个节点，将具有相同的嵌入结果。

为了进一步提高映射嵌入的质量，文章应用了一个由 Tikhonov 正则化驱动的局部细化过程，通过最小化以下目标来平滑图上的节点嵌入：

$$\min_{E_i}\left\{\left\|E_i-\widetilde{E}_i\right\|_2^2+tr(E_i{}^T L_i E_i)\right\}$$

对目标函数求导并置 0 得到：

$$E_i=(I+L_i)^{-1}\widetilde{E}_i$$

由于这种方式涉及矩阵求逆，因此以这种方式获得精确的嵌入可能效率不高。为了解决上述问题，采用更有效的频谱图过滤器来平滑嵌入，通过频域和空域变换以及数学方法，得到：

$$E_i=(\widetilde{D}_i^{-\frac{1}{2}}\widetilde{A}_i\widetilde{D}_i^{-\frac{1}{2}})^k\widetilde{E}_i=(\widetilde{D}_i^{-\frac{1}{2}}\widetilde{A}_i\widetilde{D}_i^{-\frac{1}{2}})^k H_{i+1}^i E_{i+1}$$

其中，k控制图滤波器的功效，A是邻接矩阵，D是度矩阵，$\widetilde{A}=A+\sigma I$，σ是一个很小的值确保每个节点都有一个自环，最后使用低通图滤波器来平滑映射得嵌入矩阵。

通过迭代得运用上式得到原始图的嵌入。

实验对 GraphZoom 框架与几种最先进的无监督图嵌入方法和多级嵌入框架基于 5 个标准图数据集进行了评估。实验中用到的数据集下图所示。基准方法为两种模型 DeepWalk、node2vec 和两个多层次的图嵌入框架 HARP 和 MILE。

表 1 数据集

Dataset	Type	Task	Node	Edges	Classes	Features
Cora	Citation	Transductive	2708	5429	7	1433
Citeseer	Citation	Transductive	3327	4732	6	3703
Pubmed	Citation	Transductive	19717	44338	3	500
PPI	Biology	Inductive	14755	222055	121	50
Reddit	Social	Inductive	232965	57307946	210	5414
Friendster	Social	Transductive	7944949	4466733688	5000	N/A

下图展示了分类任务的平均分类准确率，以及所有基线方法和 GraphZoom 的执行时间，关于超参数，文章使用 10 次游走，游走深度为 80，窗口大小为 10，DeepWalk 和 Node2vec 的嵌入维度均为 128。

从结果中可以看到，对于直推式学习任务，GraphZoom 可以使 DeepWalk 和

node2vec 在 Cora、Pubmed 和 Citeseer 上的分类准确率分别提高 8.3%、10.4%和 19.4%，同时达到减少 40.8 倍运行时间的目的。当增加粗化级别时，GraphZoom 仍可以以较短的 CPU 时间产生可比的甚至更好的嵌入精度。实验结果表明 GraphZoom 与底层的嵌入方法无关，能够提高各种数据集上最新的监督式嵌入方法的准确性和速度。这些结果表明，多级频谱方法可以提高填充的速度和质量，所以 GraphZoom 的运行速度要快得多。除了减小图形尺寸之外，作者的粗化方法还可以从原始图形中过滤出多余的信息，同时保留嵌入的关键频谱特性，因此分类准确率方面的嵌入质量也得到提高。

	Cora		Citeseer		Pubmed	
Method	Accuracy(%)	Time(secs)	Accuracy(%)	Time(secs)	Accuracy(%)	Time(mins)
DeepWalk	71.4	97.8	47.0	120.0	69.9	14.1
HARP(DW)	71.3	296.7 (0.3×)	43.2	272.4 (0.4×)	70.6	33.9 (0.4×)
MILE(DW, *l*=1)	71.9	68.7 (1.4×)	46.5	53.7 (2.2×)	69.6	7.0 (2.0×)
MILE(DW, *l*=2)	71.3	30.9 (3.2×)	47.3	22.5 (5.3×)	66.7	4.4 (2.3×)
MILE(DW, *l*=3)	70.6	15.9 (6.1×)	47.1	9.9 (12.1×)	64.5	2.5 (5.8×)
GZoom_F+MILE(DW)	73.8	70.6 (1.4×)	48.9	24.7 (4.9×)	72.1	7.0 (2.0×)
GZoom(DW, *l*=1)	76.9	39.6 (2.5×)	49.7	19.6 (2.1×)	75.3	4.0 (3.6×)
GZoom(DW, *l*=2)	**77.3**	15.6 (6.3×)	**50.8**	6.7 (6.0×)	75.9	1.7 (8.3×)
GZoom(DW, *l*=3)	75.1	**2.4 (40.8×)**	49.5	**1.3 (30.8×)**	**77.2**	**0.6 (23.5×)**
node2vec	71.5	119.7	45.8	126.9	71.3	15.6
HARP(N2V)	72.3	171.0 (0.7×)	44.8	174.3 (0.7×)	70.1	46.1 (0.3×)
MILE(N2V, *l*=1)	72.1	57.3 (2.1×)	46.1	60.9 (2.1×)	70.8	7.3 (2.1×)
MILE(N2V, *l*=2)	71.8	30.0 (4.0×)	45.7	28.8 (4.4×)	67.3	4.3 (3.6×)
MILE(N2V, *l*=3)	68.5	16.5 (7.2×)	45.2	15.6 (8.1×)	61.8	1.8 (8.0×)
GZoom_F+MILE(N2V)	74.3	59.2 (2.0×)	48.3	62.3 (2.0×)	72.9	7.3 (2.1×)
GZoom(N2V, *l*=1)	**77.3**	43.5 (2.8×)	**54.7**	38.1 (3.3×)	77.0	3.0 (5.2×)
GZoom(N2V, *l*=2)	77.0	13.5 (8.9×)	51.7	15.3 (8.3×)	**77.8**	1.5 (10.4×)
GZoom(N2V, *l*=3)	75.3	**3.0 (39.9×)**	50.7	**4.5 (28.2×)**	77.4	**0.4 (39.0×)**

复现结果如下图表所示：

表 2 复现结果

methed	Cora		Citeseer	
	Accuracy(%)	Time(s)	Accuracy(%)	Time(s)
Deepwalk	71.5	42.656	46.3	44.098
GZoom(DW,l=1)	75.1	41.998	47.7	39.046
GZoom(DW,l=2)	76	34.583	48.5	37.873
GZoom(DW,l=3)	74.2	28.282	50.9	32.124
node2vec	72.6	10.230	47.1	9.6234
GZoom(N2V,l=1)	75.8	15.363	49.6	11.646
GZoom(N2V,l=2)	75.3	9.867	50.3	9.635
GZoom(N2V,l=3)	74.6	6.722	49.1	7.891

复现实验采用的数据集为Cora和Citeseer，基线方法为DeepWalk和node2vec，实验对比了所有基线方法和 GraphZoom 在节点分类任务上的平均准确率以及平均执行时间。关于 DeepWalk 和 node2vec 的超参数，我们使用 10 次游走，游走

深度为 80，窗口大小为 10，DeepWalk 和 Node2vec 的嵌入维度均为 128。

从复现结果中可以看出，当粗化级别为 3 时，GraphZoom 的表现最好，GraphZoom 可以使 DeepWalk 在 Cora 和 Citeseer 上的分类准确率分别提高 6.2％、9.9％，同时运行时间能缩短 33.6%、27.15%，使 node2vec 在 Cora 和 Citeseer 上的分类准确率分别提高 4.4％、6.3％，同时运行时间能缩短 32.7%、18.00%。当增加粗化级别时，GraphZoom 可以缩短运行时间，而且也能产生优于基线方法的甚至更好的嵌入精度。这些结果表明，多级频谱方法可以提高填充的速度和质量，所以 GraphZoom 的运行速度要快得多。除了减小图形尺寸之外，作者的粗化方法还可以从原始图形中过滤出多余的信息，同时保留嵌入的关键频谱特性，因此分类准确率方面的嵌入质量也得到提高。