# APPENDIX

## A RANDOM 1-HOP SUBGRAPH DIFFUSION

We apply a random 1-hop diffusion for $S_k$. We perform a first-order Breadth-First Search (BFS) on nodes contained in $S_k$ to get the set of all 1-hop neighbors. Then we randomly sample nodes in the neighbor set and merge them into $S_k$. The number of samples is determined by the average number of nodes in the subgraph in each dataset. The purpose of diffusing subgraphs is to randomly include their external nodes to alleviate the mentioned "bias" problem in Section 1 to some extend.

Algorithm 1 summarizes the random 1-hop subgraph diffusion, where BFS($v$) means first-order Breadth-First Search (BFS) on a single node $v$, and Sample($V_N$,$K$) means randomly sample $K$ nodes in set $V_N$.

**Algorithm 1:** Random 1-hop diffusion algorithm

**Input:** Subgraph batch $\mathcal{S}$, sample number $K$, Base graph $\mathcal{G} = (V, E)$
**Output:** Diffused subgraph batch $\mathcal{S}$
**Data:** Neighbor nodes set $V_N$

**for** *i=1,2,$\cdots$, length*($\mathcal{S}$) **do**
  $V_N = \emptyset$;
  **for** $v \in V_i$ `// Every Node in the i-th Subgraph` **do**
    $V_N = V_N \cup BFS(v)$; `// 1-hop BFS`
  **end**
  `// Randomly Sample K Neighbor Nodes`
  $V_N = V_N \setminus V_i$ ;
  $V_N = Sample(V_N, K)$;
  $V_i = V_i \cup V_N$; `// Append Sampled Nodes`
  *Re-sample subgraph* $\mathcal{S}_i$*'s adjacency matrix with* $V_i$;
**end**
**return** $\mathcal{S}$

## B TRAINING PIPELINE OF PADEL

The learning ithm of PADEL is summarized as Algorithm 2. The training pipeline is divided into three parts: Self-Supervised Subgraph Augmentation, Generative Contrastive Learning, and Classification. During self-supervised subgraph augmentation, we pre-train our VSubGAE model and apply the random 1-hop subgraph diffusion as Algorithm 1. The pre-trained VSubGAE is then applied to our generative contrastive learning as an automatic and adaptive subgraph generator. In each training batch, every subgraph is treated as a positive sample once while others are treated as negative ones. Exploratory and exploitable viewed negative subgraphs are sampled for each positive sample separately in each training step. Lastly, we fine-tune our pre-trained node embedding $[\boldsymbol{X}, \boldsymbol{P}]$ and pooling models on the training set with a supervised classification task.

**Algorithm 2:** The overall pipeline of PADEL

**Input:** Dataset, batch size $K$, number of epochs $N$, subgraph generator VSubGAE, Pooling model $\mathcal{P}ooling$, Hops of RandomWalk *hop*
**Data:** Subgraph batch $\mathcal{S}$ with node neighbor embedding matrix $\boldsymbol{X}$, position embedding $\boldsymbol{P}$ and adjacency matrix batch $\mathcal{A}$

`// Self-Supervised Subgraph Augmentation`
**for** *n=1,2,$\cdots$, N* **do**
  **for** $\mathcal{S}$ *in Dataset* **do**
    $\mathcal{S} = Diffuse(\mathcal{S})$; `// Algorithm 1`
    $\boldsymbol{Z} = Eq.(7)$; `// VSubGAE Inference`
    $p(\mathcal{A}|\boldsymbol{Z}) = Eq.(8)$; `// VSubGAE Generation`
    $Loss = Eq.(9)$; `//` $\mathcal{L}_{\mathtt{VSubGAE}}$
  **end**
**end**
`// Generative Contrastive Learning`
**for** *n=1,2,$\cdots$, N* **do**
  **for** $\mathcal{S}$ *in Dataset* **do**
    $Loss = 0$;
    $\mathcal{A}^{\text{aug}} = VSubGAE([\boldsymbol{X}, \boldsymbol{P}], \mathcal{A})$; `// Exploit-View`
    $\boldsymbol{E} = \mathcal{P}ooling(\boldsymbol{X}, \boldsymbol{P}, \mathcal{A})$;
    $\boldsymbol{E}^{\text{aug}} = \mathcal{P}ooling(\boldsymbol{X}, \boldsymbol{P}, \mathcal{A}^{\text{aug}})$;
    **for** $k$ *= 1 to K* **do**
      *Treat k-th sample as positive, others the negative;*
      $\mathcal{A}^{\text{ran}} = Eq.(10)$; `// Explore-View`
      $\boldsymbol{E}^{\text{ran}} = \mathcal{P}ooling(\boldsymbol{X}, \boldsymbol{P}, \mathcal{A}^{\text{ran}})$;
      $Loss += Eq.(12)$; `//` $\mathcal{L}_{\mathtt{InfoNCE}}(k, \boldsymbol{E}^{\text{ran}}, \boldsymbol{E}^{\text{aug}})$
    **end**
  **end**
**end**
`// Classification`
**for** *n=1,2,$\cdots$, N* **do**
  **for** $\mathcal{S}$ *in Dataset* **do**
    $\boldsymbol{E} = \mathcal{P}ooling(\boldsymbol{X}, \boldsymbol{P}, \mathcal{A})$;
    $\hat{\boldsymbol{Y}} = Eq.(16)$; `// Classification`
    $Loss = Eq.(17)$; `//` $\mathcal{L}_{\mathtt{CE}}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$
  **end**
**end**