A THEOREM 3.1 AND PROOF

In this section, we provide proof for Theorem 3.1.

Theorem 3.1. *Introducing a variational approximation Q(C), a lower bound for the GIB objective is as follows:*

$$I(C;Y) - \beta I(C;G) \geq \mathbb{E}\left[\log P(Y|G)\right] + H(Y) - \boldsymbol{KL}\left(P(Y|G)\|P(Y|C)\right) \\ - \beta\mathbb{E}[\boldsymbol{KL}(P_g(C|G,Z)\|Q(C))] + \beta\mathbb{E}\left[\log\left(P_g(C|G,Z)\right)\right] + \beta H(C|G). \tag{16}$$

Proof. Specifically, we aim to transform or provide a lower bound for each of the three terms in

$$I(C;Y) - \beta I(C;G) = I(C;Y) - \beta I(C;G,Z) + \beta I(C;Z|G). \tag{17}$$

We provide the detailed deviation as follows:

$$\begin{aligned} I(C;Y) &= \mathbb{E}_{C,Y}\left[\log\frac{P(Y|C)}{P(Y)}\right] \\ &= \mathbb{E}\left[\log\frac{P(Y|G)}{P(Y)}\right] - \mathbf{KL}\left(P(Y|G)\|P(Y|C)\right) \\ &= \mathbb{E}\left[\log P(Y|G)\right] + H(Y) - \mathbf{KL}\left(P(Y|G)\|P(Y|C)\right). \end{aligned} \tag{18}$$

$$\begin{aligned} I(C;Z|G) &= \mathbb{E}_{C,Z,G}\left[\log\frac{P(C,Z|G)}{P(C|G)P(Z|G)}\right] \\ &= \mathbb{E}_{C,Z,G}\left[\log\frac{P(C|G,Z)}{P(C|G)}\right] \\ &= \mathbb{E}_{C,Z,G}\left[\log\left(P(C|G,Z)\right)\right] + H(C|G) \end{aligned} \tag{19}$$

For the following derivation, we introduce a variational approximation $Q(C)$:

$$\begin{aligned} -I(C;G,Z) &= -\mathbb{E}_{C,G}\left[\log\left(\frac{P(C|G,Z)}{Q(C)}\right)\right] + \mathbf{KL}(P(C)\|Q(C)) \\ &\geq -\mathbb{E}_{C,G}\left[\log\left(\frac{P(C|G,Z)}{Q(C)}\right)\right] \\ &= -\mathbb{E}_G[\mathbf{KL}(P(C|G,Z)\|Q(C))]. \end{aligned} \tag{20}$$

Therefore, we can finally achieve the following results:

$$I(C;Y) = \mathbb{E}\left[\log P(Y|G)\right] + H(Y) - \mathbf{KL}\left(P(Y|G)\|P(Y|C)\right) \tag{21}$$

$$I(C;G,Z) \leq \mathbb{E}_G[\mathbf{KL}(P(C|G,Z)\|Q(C))]. \tag{22}$$

$$I(C;Z|G) = \mathbb{E}_{C,Z,G}\left[\log\left(P(C|G,Z)\right)\right] + H(C|G). \tag{23}$$

Combing the above three equations (or inequations), we can achieve the final results:

$$I(C;Y) - \beta I(C;G) \geq \mathbb{E}\left[\log P(Y|G)\right] + H(Y) - \mathbf{KL}\left(P(Y|G)\|P(Y|C)\right) \\ - \beta\mathbb{E}[\mathbf{KL}(P_g(C|G,Z)\|Q(C))] + \beta\mathbb{E}\left[\log\left(P_g(C|G,Z)\right)\right] + \beta H(C|G). \tag{24}$$

□

B THEOREM 3.2 AND PROOF

In this section, we provide proof for Theorem 3.2.

Theorem 3.2. *By deriving the evidence lower bound (ELBO) for $P(C|G)$, the VGAE objective for constrained variational generation (CVG), with a hyper-parameter λ, is as follows:*

$$\max\ \mathbb{E}_Q[\log P(C|G,Z)] - \boldsymbol{KL}(Q(Z)\|P(Z|G)) - \lambda\boldsymbol{KL}\left(P(Y|G)\|P(Y|C)\right). \tag{25}$$

Proof. We first derive an evidence lower bound (ELBO) for the VGAE objective $P(C|G)$:

$$
\begin{aligned}
&\log P(C|G) \\
&= \log \int_{\mathbf{Z}} P(C, \mathbf{Z}|G) d\mathbf{Z} \\
&= \log \int_{\mathbf{Z}} Q(\mathbf{Z}|G) \frac{P(C, \mathbf{Z}|G)}{Q(\mathbf{Z}|G)} d\mathbf{Z} \\
&\quad (\textit{using Jensen's Inequality}) \\
&\geq \int_{\mathbf{Z}} Q(\mathbf{Z}) \log \frac{P(C, \mathbf{Z}|G)}{Q(\mathbf{Z})} d\mathbf{Z} \\
&= \mathbb{E}_Q[\log \frac{P(C, \mathbf{Z}|G)}{Q(\mathbf{Z})}] \\
&\quad (\textit{using the property of conditional probabilities}) \\
&= \mathbb{E}_Q[\log \frac{P(C|\mathbf{Z}, G) \cdot P(\mathbf{Z}|G)}{Q(\mathbf{Z})}] \\
&= \mathbb{E}_Q[\log P(C|\mathbf{Z}, G)] - \mathbb{E}_Q[\log \frac{Q(\mathbf{Z})}{P(\mathbf{Z}|G)}] \\
&\quad (\textit{using the definition of KL-divergence}) \\
&= \mathbb{E}_Q[\log P(C|G, \mathbf{Z})] - \mathrm{KL}(Q(\mathbf{Z}) \| P(\mathbf{Z}|G))
\end{aligned}
\tag{26}
$$

In this manner, we can optimize the VGAE by maximizing the derived ELBO. Since we aim to perform constrained variational generation, we still need to add the constraint from the supervision of class labels. Therefore, by introducing λ to control the importance of the constraint, the CVG objective with the VGAE architecture can be formulated as follows:

$$
\max \ \ \mathbb{E}_Q[\log P(C|G, Z)] - \mathrm{KL}(Q(Z) \| P(Z|G)) - \lambda \mathrm{KL}\left(P(Y|G) \| P(Y|C)\right). \tag{27}
$$

□

C Experimental Settings

In this section, we introduce the detailed settings and the datasets used in our experiments. Our code is provided at https://github.com/AnonymousSubmissionPaper/CVG.

C.1 Dataset Statistics and Details

In this subsection, we introduce the detailed statistics and the creation process for each dataset. Specifically, for the generation process of SP-Motif, MNIST-75sp, Graph-SST2, and Molhiv, we follow the settings in DIR (Wu et al., 2022c) to keep consistency. We create a novel dataset SP-Motif-Cor following the idea of SP-Motif.

- **SP-Motif** (Spurious-Motif) (Ying et al., 2019): This is a synthetic dataset that consists of 18,000 graphs. Each graph is composed of a base (Tree, Ladder, Wheel denoted by $S = 0, 1, 2,$, respectively) and a motif (Cycle, House, Crane denoted by $C = 0, 1, 2,$, respectively). The true label Y is solely determined by the motif C. In the training set, we introduce false relations of varying degrees between the base S and the label Y. Specifically, each motif is sampled from a uniform distribution, while the distribution of its base is determined by $P(S) = b \times \mathbb{I}(S = C) + (1 - b)/2 \times \mathbb{I}(S \neq C)$. We manipulate the parameter b to create Spurious-Motif datasets with distinct biases. In the testing set, motifs, and bases are randomly attached to each other, and we include graphs with large bases to magnify the distribution gaps.
- **MNIST-75sp** (Knyazev et al., 2019): This dataset converts MNIST images into 70,000 superpixel graphs, with each graph containing at most 75 nodes. Superpixels represent nodes, while edges denote the spatial distance between the nodes. Each graph is labeled into one of 10 classes, and random noises are added to nodes' features in the testing set.

- **Graph-SST2** (Socher et al., 2013; Yuan et al., 2022): Graph-SST2 comprises graphs labeled according to sentence sentiment. More specifically, nodes represent tokens, and edges indicate node relations. Graphs are partitioned into different sets based on their average node degree to induce dataset shifts.
- **Molhiv** (OGBG-Molhiv) (Wu et al., 2018; Hu et al., 2020; 2021): This dataset is designed for molecular property prediction and contains molecule graphs. Specifically, nodes represent atoms, and edges represent chemical bonds. Each graph is labeled based on whether a molecule inhibits HIV replication or not.
- **SP-Motif-Cor**: We create this synthetic dataset to manually inject various degrees of intra-graph correlations for evaluation. Specifically, we keep the same number of graphs in different subsets as SP-Motif. To inject correlations for each graph, given a defined bias degree b, we set the features of 50% randomly selected nodes on its base graph (i.e., Tree, Ladder, or Wheel) to the same values as Y (i.e., 0, 1, 2). For others, the probability is set to $(1-b)/2$ for the other two labels. We only alter the features on 50% nodes to avoid the model naively learning classification from averaging all node features. Note that the features are only injected into the base graph. Therefore, existing methods that extract invariant subgraphs (i.e., motifs in this case) inevitably involve nodes in the base graph, which is spurious for classification based on the motif. As a result, extracting such correlated and spurious nodes will adversely impact the classification performance based on motifs.

C.2 Basic Settings

In this subsection, we introduce the basic settings in our experiments.

Backbone Settings. Specifically, the classifier $f(\cdot)$ in our framework consists of a GNN and an MLP. For the GNN, we leverage a 3-layer GCN (Kipf & Welling, 2016) with a hidden size of 128, except for SP-Motif, on which we use a hidden size of 64. The dropout rate on all datasets is 0.1. The GNN is followed by an MLP to calculate the final prediction for the specific label of a graph. For the encoder GNN in our generator $g(\cdot)$, we follow the setting sin DIR (Wu et al., 2022c) and use various GNNs for different datasets. The activation functions are all set as the ReLU function.

Training Settings. For the training, we conduct all experiments on a NVIDIA A6000 GPU with 48GB memory. We utilize the batched GNN pipeline and set the batch size as 32 for all datasets. The learning rate is set as 10^{-3}. For optimization, we use the Adam optimizer (Kingma & Ba, 2015) with a weight decay rate of 0. We follow the specific dataset split used in DIR. For all datasets, we train our framework for 500 epochs. We report the performance of the model obtained on the epoch that achieves the best validation performance. All the experiments are implemented with PyTorch (Paszke et al., 2019) under a BSD-style license. The required packages are listed below.

- Python == 3.7.10
- torch == 1.8.1
- torch-cluster == 1.5.9
- torch-scatter == 2.0.6
- torch-sparse == 0.6.9
- torch-geometric == 1.4.1
- numpy == 1.18.5
- scipy == 1.5.3

C.3 Hyper-Parameter Settings

In this section, we introduce the detailed parameter settings for our experiments. Specifically, we set the number of nodes in C as $N = 5$. For the regularization loss weight β_r and the generation loss β_g, we both set them as 1. The ϵ in the regularization loss is set as 0.7 for SP-Motif and 0.5 for other datasets. The number of positive samples S and the number of negative samples K in the generation loss are set as 1 and 5, respectively. The temperature is set as 1.