Algorithm 1: Training algorithm for explaining graph classification

Input: A set of input graphs (normalized $V \times V$ symmetric matrix observations) with i-th graph represented by $G_o^{(i)}$, node features $\mathbf{X}^{(i)}$ (if available), graph features $\mathbf{H}^{(i)}$, and a binary outcome $Y^{(i)}$, a trained GNN model: $\mathrm{GNNE}_{\Phi_0}(\cdot)$ and $\mathrm{GNNC}_{\Phi_1}(\cdot)$.

```
for each graph $G_o^{(i)}$ do
    $\mathbf{Z}^{(i)} \leftarrow \mathrm{GNNE}_{\Phi_0}(G_o^{(i)}, \mathbf{X}^{(i)})$.
    $Y_o^{(i)} \leftarrow \mathrm{GNNC}_{\Phi_1}(\mathbf{Z}^{(i)})$.
end for
for each epoch do
    for each graph $G_o^{(i)}$ do
        $\Omega \leftarrow$ latent variables parameterized by neural networks.
        for $k \leftarrow 1$ to $K$ do
            $\hat{G}_s^{(i,k)} \leftarrow$ sampled from binary edge variables (preserving weights if needed).
            $\hat{Y}_s^{(i,k)} \leftarrow \mathrm{GNNC}_{\Phi_1}(\mathrm{GNNE}_{\Phi_0}(\hat{G}_s^{(i,k)}, \mathbf{X}^{(i)}))$
        end for
    end for
    Compute loss with the objective function.
    Update parameters $\Psi$ with backpropagation.
end for
```