Symmetry-induced Disentanglement on Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning disentangled representations is important for unraveling the underlying complex interactions between latent generative factors. Disentanglement has been formalized using a symmetry-centric notion for unstructured spaces, however, graphs have eluded a similarly rigorous treatment. We fill this gap with a new notion of conditional symmetry based disentanglement, and leverage tools from Lie algebras to encode graph properties into subgroups using suitable adaptations of generative models such as Variational Autoencoders. Unlike existing works on disentanglement, proposed models can learn to segregate the latent space into uncoupled and entangled parts. Experiments on synthetic and real datasets reveal the ability of these models to learn effective disengaged representations, and improve performance on downstream tasks such as few-shot classification and molecular generation.

1 Introduction

Disentanglement represents a fundamental desideratum in Artificial Intelligence, because it captures information about the salient (or explanatory) factors of variation in the data, and it isolates information about each specific factor in only a few dimensions, unraveling the underlying interactions of complex data [2, 56]. Disentangled representations have often been deemed responsible for providing neural models with the ability to improve real-word downstream tasks [2, 55, 17, 56, 32], and have demonstrated empirically the ability to enable critical properties such as sample efficiency [18] and systematic generalization [65, 28, 39].

Higgins et al. [17] have recently presented a symmetry-based definition for disentanglement, which uses tools from group theory to formalize the idea that disentanglement is achieved when the abstract and the vector spaces are equivariant. Such symmetry-based definition presents various advantages: first it aligns with the idea that auxiliary information can be exploited for imposing a structure on the latent space [21, 36], secondly, it has been successfully used as a theoretical framework for recent disentangled Variational Autoencoder (VAE) works [23, 47] [60, 66], finally, differently from other definitions of disentanglement [10, 12, 2] it may enable a fundamental connection with several graph neural network (GNN) approaches that leverage group symmetries [5, 35, 9, 3].

Disentanglement approaches have been initially developed on vision-based datasets [16, 6, 22, 8, 14], however, numerous recent methods have improved GNNs performance using disentanglement [11, 37, 34, 27, 61]. For example, [34] propose to use a convolutional layer to identify the factor responsible for linking a node to its neighbors via a neighborhood routing mechanism, while [61] enable multi-relational disentanglement, by encoding all the possible links between nodes with attention.

While previous work in the GNN domain has demonstrated the advantages of exploiting group symmetries [35, 3], investigating disentanglement from a symmetry group perspective is still an unexplored area in the GNN literature. Furthermore, a rigorous formalization of disentanglement for

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

graphs, which aligns the formalism provided by [17] within a GNN architecture is still missing in the contemporary disentangled GNN literature [34, 61, 27].

In this paper, we make several contributions to fill the mentioned gaps in disentangled GNNs.

- We introduce two formal definitions for unconditional and conditional symmetry-based disentanglement on graphs, which provide the tools for encoding graph properties into subgroups by leveraging a Lie algebra parameterization. Our intuition is that each generative factor can correspond to a Lie algebra coordinate element, and control a single graph property (Section 2).
- Based on the proposed Lie group disentanglement framework, we design two disentangled GNN modules (unconditional and conditional) that auto-encode both a graph representation and a Lie group, within a VAE framework (Section 3).
- We provide the first systematic evaluation on several disentanglement metrics [22, 16, 8, 12, 42], and demonstrate that the proposed models can outperform all contemporary GNN baselines [34, 61], and successfully learn to segregate the latent space into uncoupled and entangled parts (Section 5.1). Finally, we show that our models are able to improve performance on downstream tasks such as few-shot classification, compression (Section 5.2) and molecular graph generation (Section 5.3).

2 Proposed framework

2.1 Symmetry-based disentanglement

We consider the definition for disentangled representations from Higgins et al. [17].

Definition 1 (Symmetry-based disentanglement)**.** Let us consider a generative process $b = W \to O$ mapping a set of world states W into a set of observations (i.e. data) O, and an inference process $h = O \to Z$ mapping the data into the vector space. By composing b and h, we obtain $f = h \circ b$, which is equivalent to $f : W \to Z$. Given group actions on $W \cdot : G \times W \to W$ and on $Z \cdot : G \times Z \to Z$, the function f is defined equivariant between the actions on W and Z if the action commutes with f, which can be expressed as an equivariant map: $g \cdot f(w) = f(g \cdot w), \forall g \in G, \forall w \in W$. Assuming that the symmetry group G can be decomposed into subgroups $G_1 \times \ldots \times G_n$, a vector representation Z is disentangled with respect to the decomposition of G if: (1) there is an action defined on the representation set $\cdot : G \times Z \to Z$, (2) the map $f : W \to Z$ is equivariant between actions on W and Z, (3) there is a decomposition $Z = Z_1 \times Z_2 \ldots \times Z_n$ such that each Z_i is affected only by G_i and fixed for $G_j, j \neq i$.

Following the notation of Definition 1, we propose the notion of conditional disentanglement.

Definition 2 (Conditional symmetry-based disentanglement)**.** We consider: a generative process $b = W \times E \to O$ mapping a set of disentangle-able world states W and non-disentangle states E into a set of observations (i.e. data) O, and an inference process $h = O \to Z$ mapping the data into the vector space. By composing b and h, we obtain $f = b \circ h$, which maps $W \times E$ to Z. Given group actions on $W \cdot : G \times W \to W$ and on $Z \cdot : G \times Z \to Z$, the function f is defined equivariant between the actions on W and Z if the action commutes with f according to the following equivariant map: $g \cdot f(w, e) = f(g \cdot w, e), \forall g \in G, \forall w \in W, \forall e \in E$.

2.2 Lie algebras for graph networks

While previous works explored disentanglement on GNNs [34, 61], they lack formal frameworks for designing disentangled GNNs, which is necessary for aligning with the notions of symmetry groups from Definition 1. In the next two paragraphs, we introduce Lie algebras and leverage them to provide a disentangled GNN architecture.

Lie groups and Lie algebras A Lie group G is a group of symmetries where the symmetries are continuous [49]. A Lie group is associated with a Lie algebra $\mathfrak{g}$, which is the tangent space to the identity element of G. A Lie algebra is parameterized with a basis $\{\mathbb{A}_i\}_{i=1}^k$, where every element in $\mathfrak{g}$ can be written as $\mathbb{A} = \mathbb{A}_1 t_1 + \ldots + \mathbb{A}_k t_k$, where t_i are the coordinates. Elements of the Lie algebra can be mapped into the Lie group with an exponential function: $\exp : \mathfrak{g} \to G$. In the next paragraph,

we show how we can connect the notion of a Lie group G with the definition of disentanglement in a GNN model, by associating each Lie algebra coordinates t_i element with a generative factor.

GNN architecture Given two latent variables Z and T, we instantiate the unconditional disentanglement from Definition 3 with a probabilistic model that maximize the log-likelihood of the graph data $\mathcal{G} = (X, \mathcal{A})$, as follows:

We consider an input graph $\mathcal{G} = (X, \mathcal{A})$ where X denotes the matrix obtained by stacking the features of the nodes, and $\mathcal{A}$ denotes the adjacency matrix. Let N be the number of nodes in G and d the number of features, so $X \in \mathbb{R}^{N \times d}$. We consider a Lie group of dimension L associated with k Lie algebra basis elements. First, we compute the graph features $f = \text{Pool}(\text{GCN}(\mathcal{G})) \in \mathbb{R}^{N_1 \times L}$, where the GCN maps the features to a group matrix of size $N \times L$ and Pool reduces the number of nodes to N_1 using a certain pooling rate. Next, the second layer of the GCN maps the features f to a latent code Z of dimension $N_1 \times k$. To enhance disentanglement, Z is then fed into k GCNs, where each the output of each GCN is another $N_1 \times k$ matrix. We then perform an aggregation operation on each GCN as: $T = \text{agg}(\{\text{GCN}_j(Z) = t_j\}_{j=1}^k)$. The vector T represents the k coordinates of a Lie algebra $\mathfrak{g}$, and models the generative factors of the dataset. Finally, T is fed into the exponential map, which reconstructs the variable $\hat{Z}$, while imposing a group structure on it, by learning the basis $\mathbb{A} \in \mathfrak{g}$, i.e., $\hat{Z} = \exp \mathbb{A}(T)$ where $\mathbb{A} \in \mathfrak{g}$ and $\mathbb{A}(T) = \sum_{i=1}^k \mathbb{A}_i t_i$. This model is further explained in Section 3.

2.3 Formalizing disentanglement on graphs

Intuitive definition The latent representation $\hat{Z}$ is disentangled with respect to the Lie group G, if a change in the coordinate t_i is associated with a change in $\hat{z}_i$. In other words, disentanglement means that the semantics $(t_1, t_2, \ldots, t_k)$ are equivariant with respect to the properties that are being predicted $(\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_k)$. Next, we provide the formal definitions for unconditional and conditional graph disentanglement, where the former expresses $\hat{Z}$ in function of T without accounting for Z, while the latter expresses $\hat{Z}$ based on T, after fixing an encoding Z.

Definition 3 (Unconditional graph disentanglement)**.** The graph embedding $\hat{Z}$ obtained by $f(\hat{Z}|T)$ is unconditionally disentangled with respect to the Lie group coordinates $T = \{t_j\}_{j=1}^k$ if: (1) there is a group action on $\hat{Z}$: $\cdot : G \times \hat{Z} \to \hat{Z}$, (2) the map $f = \exp \mathbb{A}(T) : T \to \hat{Z}$ is equivariant between actions on T and $\hat{Z}$, (3) there is a decomposition $\hat{Z} = \hat{z}_1 \times \hat{z}_2 \ldots \times \hat{z}_k$, where each coordinate t_i affects only the corresponding component $\hat{z}_i$.

Definition 4 (Conditional graph disentanglement)**.** The graph embedding $\hat{Z}$ obtained by $f(\hat{Z}|T, Z)$ is conditionally disentangled with respect to the Lie group coordinates $T = \{t_j\}_{j=1}^k$ conditioned on Z if (1) there is a group action on $\hat{Z}$: $\cdot : G \times \hat{Z} \to \hat{Z}$, (2) the map $f : \exp \mathbb{A}(T) \times Z \to \hat{Z}$ is equivariant between actions on T and $\hat{Z}$ for any fixed Z, and (3) $\exp \mathbb{A}(T)$ factorizes into the product $\prod_{i=1}^k \exp(t_i \mathbb{A}_i) \times Z$ such that each component $\exp(t_i \mathbb{A}_i) \times Z$ is affected only by the corresponding coordinate t_i for $i \in \{1, 2, \ldots, k\}$.

2.4 Disentanglement constraints

Since a Lie group parameterization is not sufficient for a group to be decomposed into subgroups that are parameterized independently by a single coordinate [66, 60], we provide the necessary constraints for enforcing disentanglement.

Commutative penalty First, disentanglement can be achieved by enforcing a coordinate t_i to be identified by single group $\exp(t_i \mathbb{A}_i)$, representing a single property variation in the data $\hat{z}_i$. Such property is expressed by the equivalence between the exponential map of the sum, and the product over exponential maps, which [66] proved to be achieved when there is commutativity between the Lie algebra basis. Namely, if $\mathbb{A}_i \mathbb{A}_j = \mathbb{A}_j \mathbb{A}_i$ then $\exp \sum_{i=1}^k t_i \mathbb{A}_i = \prod_{i=1}^k \exp t_i \mathbb{A}_i$.

Hessian penalty Furthermore, disentanglement can be strengthened using the hessian penalty [42], which states that the Hessian matrix with respect to a disentangled representation is always zero, since there should be independence between different dimensions. Zhu et al. [66] adapt this penalty

in a Lie algebra parameterization setup, as $H_{ij} = \frac{\delta^2 g(T)}{\delta t_i \delta t_j}$ where $g(T) = \exp \sum_{i=1}^{k} t_i \mathbb{A}_i$, and also show that if $\mathbb{A}_i \mathbb{A}_j = 0, \forall i, j$ then $H_{ij} = 0$.

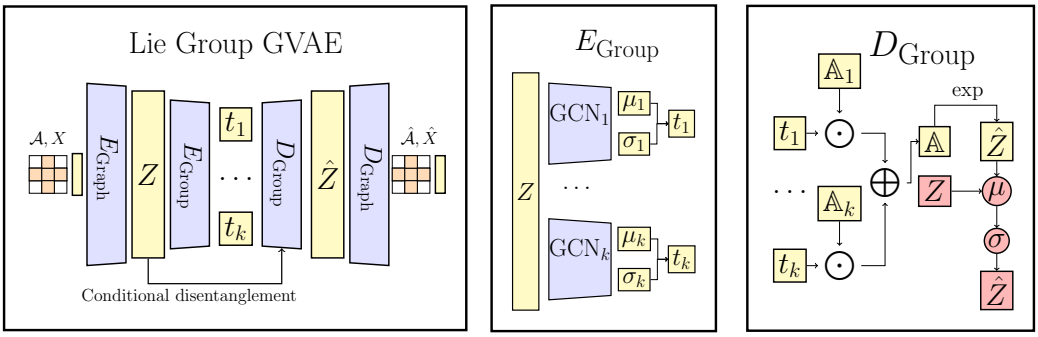

Figure 1: Lie group GVAE architecture and layers. Left panel: Lie group GVAE is composed of 4 blocks: 2 encoders, and 2 decoders for graph and group. Middle: the group encoder obtains the Lie algebra coordinates via k GCNs. Right: The group decoder combines the coordinates with basis elements to reconstruct the graph latent code $\hat{Z}$. In the conditional case (red nodes) we further condition on Z by taking the mean (μ) between Z and $\hat{Z}$ and passing it through an activation function σ (ReLu).

3 Method

We design a Lie group graph VAE framework [24], as depicted in Figure 1, which considers the latent graph encoding variable Z, the graph features X, the symmetry Lie group structure T, and the graph adjacency matrix $\mathcal{A}$. We propose two decoders, the unconditional one reconstructs the properties $\hat{Z}$ given T, while the conditional one reconstructs the properties $\hat{Z}$ given Z and T.

3.1 Probabilistic model

Proposition 1 (Unconditional ELBO)**.** Given two latent variables Z and T, we instantiate the unconditional disentanglement from Definition 3 with a probabilistic model that maximize the log-likelihood of the graph data $\mathcal{G} = (X, \mathcal{A})$, as follows:

$$\mathcal{L}_{\text{unc}} = \mathbb{E}_{q(Z|\mathcal{G})q(T|Z)} \log p(\mathcal{G}|\hat{Z})p(\hat{Z}|T) - \mathbb{E}_{q(Z|\mathcal{G})}\text{KL}(q(T|Z)||p(T)) - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \quad (1)$$

Proposition 2 (Conditional ELBO)**.** We extend Proposition 1 with a conditional version, corresponding to the conditional disentanglement from Definition 4.

$$\mathcal{L}_{\text{cond}} = \mathbb{E}_{q(Z|\mathcal{G})q(T|Z)} \log p(\mathcal{G}|\hat{Z}, Z)p(\hat{Z}|T) - \mathbb{E}_{q(Z|\mathcal{G})}\text{KL}(q(T|Z)||p(T)) - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \quad (2)$$

Propositions 1 and (2) are proved respectively in Appendix B and C.

Equations (1) and (2) are implemented with four modules, namely, two stochastic networks $q(Z|\mathcal{G})$ (E_{graph}), encoding graph $\mathcal{G}$ into vector Z and $q(T|Z)$ (E_{group}), encoding the group structure T given graph encoding Z, and two deterministic networks $p(\hat{Z}|T)$ (D_{group}), decoding $\hat{Z}$ from the group structure T, and $p(\mathcal{G}|\hat{Z})$ (D_{graph}), decoding the graph $\mathcal{G}$ from $\hat{Z}$. Note that the decoder D_{group} is responsible for the group structure injection, because it imposes $p(\hat{Z}|T)$ to be a Lie algebra parameterization. Furthermore, in the conditional ELBO model the D_{group} decoder is given by $p(\mathcal{G}|\hat{Z}, Z)$, where the reconstructed latent code $\hat{Z}$, is conditioned on the fixed latent variable Z.

3.2 Encoders

We first encode the graph representation in the latent space Z, by decomposing the features X and adjacency matrix $\mathcal{A}$ using an inference network E_{graph} $q(Z|X, \mathcal{A})$. We use a single 2-layer GCN, where the first layer reduces the graph features to the symmetry group dimension, and performs a pooling operation to reduce the number of nodes: $\overline{X} = \text{Pool}(\text{GCN}(X, \mathcal{A}))$, while the

second layer reduces the number of features to k while computing the matrices $\mu = \text{GCN}_\mu(\overline{X}, \mathcal{A})$, $\log\sigma = \text{GCN}_\sigma(\overline{X}, \mathcal{A})$, which are then combined into a Z with the reparameterization trick, as $q(Z|X, \mathcal{A}) = \mathcal{N}(Z|\mu, \text{diag}(\sigma^2))$.

Secondly, we use E_{group} to map the latent vector Z into a vector T, which represents the k Lie algebra coordinates. Each $\{t_i\}_{i=1}^k$ is the output of a graph convolutional encoder (GCN) over Z. Each GCN consists of two layers, where first one computes $\overline{Z} = \text{GCN}(Z, \mathcal{A})$, while the second one computes the matrices $\mu = \text{GCN}_\mu(\overline{Z}, \mathcal{A})$ and $\log\sigma = \text{GCN}_\sigma(\overline{Z}, \mathcal{A})$, which are then combined to compute t_i with the reparameterization trick.

$$q(T|Z, \mathcal{A}) = \prod_{i=1}^k q(t_i|Z, \mathcal{A}) \quad \text{with} \quad q(t_i|Z, \mathcal{A}) = \mathcal{N}(t_i|\mu_i, \text{diag}(\sigma_i^2)) \tag{3}$$

3.3 Decoders

We first leverage D_{group} for decoding the latent variable $\hat{Z}$ using a Lie group G and a Lie algebra $\mathfrak{g}$, with a Lie algebra parameterization. Specifically, for each coordinate $t_i \in T$, we learn a Lie algebra basis element $\{\mathbb{A}_i\}_{i=1}^k \in \mathfrak{g}$. The coordinates and basis are then aggregated and fed into an exponential mapping layer, to achieve a group representation, as follows.

$$p(\hat{Z}|T) = g(T) = \exp \mathbb{A}(T) \quad \text{where } \mathbb{A}(T) = \sum_{i=1}^k t_i \mathbb{A}_i \quad \text{for } g \in G,\ \mathbb{A} \in \mathfrak{g} \tag{4}$$

For the conditional version of the ELBO, D_{group} $\hat{Z}$ is computed by combining Z and $\mathbb{A}(T)$, as $p(\hat{Z}|Z, T) = \sigma(\text{agg}[\exp \mathbb{A}(T), Z])$, where $Z = \{z_j\}_{j=1}^k$, agg is the mean, and σ is the ReLU non-linear activation function.

Secondly, we apply D_{graph}, which first maps the latent $\hat{Z}$ into N latent variables using sigmoid functions σ_j, and then aggregates the N latent variables to obtain the full graph representation.

$$p(\mathcal{A}, X|\hat{Z}) = \prod_{j=1}^N \left[p(\mathcal{A}, X|\sigma_j(\hat{Z}))\right] \tag{5}$$

3.4 Training

The proposed unconditional and conditional models are trained by optimizing the loss given by $\beta\mathcal{L} + \lambda h + \gamma c$, where $\mathcal{L}$ is the ELBO, from Equation (1) (unconditional) or (2) (conditional), while h and c are respectively the hessian and commutative penalties, obtained as regularization terms. The parameters are found empirically, and we provide further insights on their effect on the model and its performance by providing an ablation study in the experiments in Section 5. In terms of neural architecture we set each encoder and decoder to be made of 3 of the described layers, where each layer takes as input the graph features of the previous layer.

4 Related work

Symmetry-based disentanglement Initial works on symmetry disentanglement [7, 45] have proposed use reinforcement learning to achieve irreducible representations of group elements through observation of action transitions, while [41] does not require labelled action-transition pairs. In a related line of work, [66] propose to decompose a Lie group in multiple subgroups, and enforce commutativity and independence between subgroups, while [60] impose a cyclic group structure, while encouraging such group to be isomorphic to the ground truth. Other works focus on manifolds properties, for example, [43], propose a nonparametric algorithm that aims to discover a decomposition of a data manifold by investigating its holonomy group, while [13] model disentanglement as a product of the low-dimensional submanifolds underlying the data space. Finally, [58] design a method which iteratively updates a partition, modeled as a symmetry group, until the considered factor is invariant, while [4] uses group representation theory for defining distributed latent operators. Differently, from the reviewed methods, we propose to analyze the graph data modality.

Disentangled GNNs [34] propose a routing mechanism aimed at separating the underlying factors in a graph dataset, while [61] use an attention mechanism for disentangling the representation of GNNs, and provide two evaluation metrics for disentanglement on graphs. In a related line

of work, [31] achieve disentanglement by encouraging independence in the latent space, while [1] propose a method for disentangling multi-relational graphs with an application on pedestrian trajectory prediction. Furthermore, [11] propose a generative model for spatio-temporal graphs via disentangled representations, while [27] leverage a contrastive framework for disentangling the latent factors in GNNs. Differently from the mentioned GNNs methods, we provide a formal definition of disentanglement for graphs, and do leverage group structures for modeling.

Graph generative models We also relate to VAEs models on graph data modality, for example [59, 64] propose VAE-based Information Bottleneck methods for improving GNNs performance on classification tasks, by learning minimal sufficient representations. Another line of work aims to generate molecular graphs, by proposing graph-VAE models that are provided with encoder and decoders able to handle graph structures, and to condition molecule generation on specific properties [30, 53]. Similarly, [63] propose a deep autoregressive model that learns to generate graphs by training on a representative set of graphs and decomposes the graph generation process into a sequence of node and edge formations. We differ from these methods as we explicitly investigate the disentanglement capabilities of generative models.

5 Experiments

Section 5.1 evaluates the disentanglement capabilities of the models, Section 5.2 provides a compression and few-shots classification experiment, Section 5.3 assesses the generation capabilities on molecular datasets. [1]

5.1 Disentanglement evaluation

Experimental setup Vision-based disentanglement [16, 22] requires the generative factors to be known, is based on synthetic datasets that have such property. We follow the protocol, and construct two graph datasets based respectively on random graphs, where we can sample a combination of generative factors that are required to synthesize node features and adjacency matrices in each graph. More details on the datasets are in Appendix A. We measure 5 disentanglement metrics from the literature, including βVAE metric [16] (Beta), FactorVAE metric (FVM) [22], Mutual Information Gap (MIG) [8], DCI Disentanglement [12], and Factor Leakage (FL) [41].

Ablation study We start by evaluating the impact of our model components on the disentanglement scores, by maintaining all parameters fixed, except from one that varies. We probe group size, hessian penalty, commutative penalty, subspace size, beta penalty, and model depth. In Figure 2 we report the mean over 10 runs for our unconditional model on the synthetic Watts-Strogatz dataset. We observe that when the group size is increased, there is an improvement in disentanglement across all metrics, and this can be explained by the idea that with a larger group size, the subgroups, represented by the Lie algebra coordinate t_i can control different data variations. The hessian penalty results to be effectively enhancing disentangled representations, while the commutativity penalty is less effective, and we can explain this result, because the former requires the Lie algebra basis elements to have mutual products of zeros while subgroup decomposition only requires their commutators to be zeros, which also confirms the results obtained for image datasets [66].

Quantitative disentanglement benchmark We compare our models with several baselines, over 10 random seed runs, where we measure the mean and variance, by training for 200 epochs on the proposed synthetic datasets and measuring the quantitative metrics on 1000 data points. Baselines include GAT [57] and GCN [25] based autoencoders, as well as disentangled graph methods such as FAC [61] and DIS [34]. Our unconditional and conditional Lie group models are named UNC and CON and set with group size of 81, a hessian penalty of 40, and a commutativity penalty of 5.

In Table 1 we observe that our Lie group models are able to improve on previous baselines on all metrics on the evaluated datasets, and the conditional version achieves the best performance in most cases. The FL and FVM metrics show the highest performance increase. FL is designed to capture the disentanglement on each generative factor, which, in our models is controlled by the group size, while in previous approaches is not considered. FVM is the most used metrics for vision datasets,

[1] Code is provided as zip file under Apache license. We use 2-4 NVIDIA Tesla V100 GPUs for training.

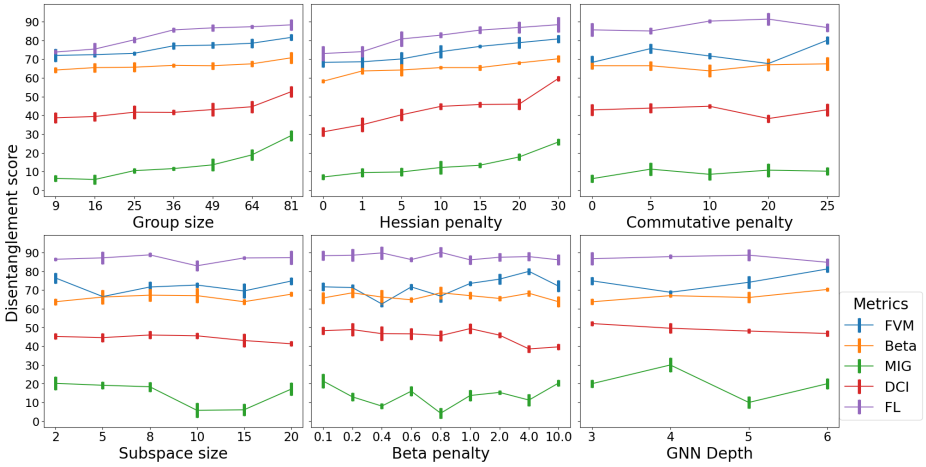

Figure 2: Impact of single model components on disentanglement metrics.

Table 1: Disentanglement results.

	Watts-Strogatz					Erdos-Renyi				
Model	FVM	Beta	MIG	DCI	FL	FVM	Beta	MIG	DCI	FL
GCN	$69.6_{\pm 0.8}$	$66.5_{\pm 0.7}$	$23.2_{\pm 2.5}$	$43.8_{\pm 1.6}$	$84.1_{\pm 0.5}$	$57.0_{\pm 4.5}$	$61.0_{\pm 1.4}$	$22.5_{\pm 3.3}$	$54.1_{\pm 2.1}$	$71.7_{\pm 2.6}$
GAT	$72.6_{\pm 1.3}$	$66.2_{\pm 3.4}$	$17.4_{\pm 1.0}$	$53.2_{\pm 2.7}$	$79.0_{\pm 1.1}$	$56.0_{\pm 3.2}$	$66.3_{\pm 2.5}$	$31.4_{\pm 1.9}$	$54.0_{\pm 2.9}$	$70.2_{\pm 2.1}$
FAC	$74.2_{\pm 1.3}$	$65.7_{\pm 2.6}$	$26.4_{\pm 2.0}$	$53.4_{\pm 4.0}$	$80.3_{\pm 0.9}$	$67.2_{\pm 1.7}$	$66.3_{\pm 3.3}$	$35.1_{\pm 2.4}$	$53.8_{\pm 1.5}$	$78.1_{\pm 3.0}$
DIS	$73.0_{\pm 3.5}$	$69.3_{\pm 2.1}$	$27.1_{\pm 1.7}$	$52.2_{\pm 3.9}$	$85.2_{\pm 2.0}$	$68.1_{\pm 1.8}$	$64.1_{\pm 0.5}$	$41.3_{\pm 1.4}$	$53.5_{\pm 2.3}$	$72.4_{\pm 2.3}$
UNC	$80.5_{\pm 1.5}$	$71.6_{\pm 4.5}$	$28.5_{\pm 2.8}$	$55.4_{\pm 2.0}$	$91.0_{\pm 2.3}$	$83.0_{\pm 2.9}$	$71.8_{\pm 3.2}$	$\mathbf{53.2}_{\pm 0.8}$	$58.6_{\pm 1.9}$	$87.1_{\pm 4.0}$
CON	$\mathbf{81.9}_{\pm 1.2}$	$\mathbf{74.5}_{\pm 3.2}$	$\mathbf{49.1}_{\pm 3.1}$	$\mathbf{59.7}_{\pm 1.2}$	$\mathbf{92.3}_{\pm 0.9}$	$\mathbf{83.3}_{\pm 0.7}$	$\mathbf{74.1}_{\pm 1.2}$	$52.4_{\pm 3.1}$	$\mathbf{60.5}_{\pm 1.2}$	$\mathbf{89.8}_{\pm 2.3}$

and provides a more robust assessment than the Beta metric. We note that while previously proposed disentangled approaches DIS and FAC demonstrate higher disentanglement when compared to GCN and GAT, they are outperformed by our methods.

5.2 Compression and few-shots classification

Experimental setup Following the setup from [15], we consider the task of learning a compressed model via multiple pooling operations, and test on few-shots classification capabilities on datasets with multiple classes, including FRANKENSTEIN [40] (2 classes), COLORS-3 [26] (11 classes), Mutagenicity [48] (2 classes), NCI1 [51] (2 c). The baselines include MIA [15], which propose an attention-based pooling for compression, as well as GCN [25], and GAT [57] based autoencoders, equipped with the same pooling layer. All models are set to have a pooling rate of 0.8, and a depth of 3. Our models are set with a group size of 81, hessian penalty of 20 and commutativity penalty of 5. The task consists in training first without labels to learn the representation, and then using the compressed model as input for training a GCN classifier a labelled subset of the dataset, which involves 100 samples for training and 100 for testing.

Results In Table 2 we report the mean and variance across 10 runs for MSE, the classification accuracy, and the size of the trained model. The results show that CON and UNC outperform the considered baselines in terms of few-shots classification accuracy on all datasets, while achieving a smaller compressed size. We hypothesize that the generalization performance is due to more efficient learning due to disentanglement, in fact disentangled representations have been studied in the context of generalization in previous work [38, 65, 28]. In terms of MSE, our models show a decrease in performance compared to MIA, but improve on GCN and GAT based autoencoders.

Table 2: Compression and few shot classification results.

	FRANKENSTEIN			COLORS-3			Mutagenicity			NCI1		
Model	MSE	Size	Acc.	MSE	Size	Acc.	MSE	Size	Acc.	MSE	Size	Acc.
MIA	**1.6**$_{\pm 0.5}$	4.9M	65$_{\pm 1}$	**1.8**$_{\pm 0.2}$	729K	33$_{\pm 2}$	4.1$_{\pm 0.8}$	753K	78$_{\pm 2}$	**3.2**$_{\pm 0.5}$	777K	61$_{\pm 1}$
GCN	12.6$_{\pm 2.5}$	4.2M	39$_{\pm 3}$	11.2$_{\pm 1.1}$	657K	33$_{\pm 2}$	16.5$_{\pm 2.1}$	673K	72$_{\pm 1}$	7.4$_{\pm 1.2}$	697K	53$_{\pm 4}$
GAT	9.0$_{\pm 1.2}$	4.2M	26$_{\pm 2}$	10.1$_{\pm 3.1}$	665K	30$_{\pm 1}$	11.3$_{\pm 2.8}$	681K	79$_{\pm 4}$	8.1$_{\pm 0.3}$	705K	50$_{\pm 2}$
UNC	4.3$_{\pm 1.1}$	**4.1M**	68$_{\pm 2}$	5.1$_{\pm 1.5}$	**625K**	**39**$_{\pm 1}$	4.2$_{\pm 0.3}$	**633K**	81$_{\pm 3}$	3.5$_{\pm 0.3}$	**657K**	63$_{\pm 2}$
CON	2.8$_{\pm 0.5}$	**4.1M**	**69**$_{\pm 2}$	1.9$_{\pm 0.4}$	**625K**	38$_{\pm 3}$	**3.2**$_{\pm 1.1}$	**633K**	**84**$_{\pm 2}$	3.9$_{\pm 0.9}$	**657K**	**66**$_{\pm 2}$

Table 3: Random graph generation.

	ZINC				QM9				MOSES			
Model	Val	Res	Uni	Nov	Val	Res	Uni	Nov	Val	Res	Uni	Nov
JT-VAE	**100**	n/a	**100**	**100**	n/a	n/a	n/a	n/a	**100**	n/a	**99.96**	91.43
GCPN	**100**	20	99.97	**100**	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
GraphAF	**100**	68	99.1	**100**	**100**	67	94.15	88.83	**100**	71	99.99	**100**
GraphDF	**100**	**89.03**	99.16	**100**	**100**	82.67	**97.62**	**98.1**	**100**	**87.58**	99.55	**100**
UNC	**100**	78.21	99.12	**100**	**100**	75.43	95.22	90.33	**100**	81.62	99.62	**100**
CON	**100**	86.02	99.14	**100**	**100**	**82.92**	96.43	92.28	**100**	84.29	99.79	**100**

5.3 Molecular graph generation

The molecular generation experiment is two-fold, the first part involves learning the distribution of the molecular data while the second one involves fine-tuning the pre-trained model with a Reinforcement Learning (RL) procedure for a downstream generation task. We employ our disentangled models during the training phase, and then follow the RL method from GCPN [62] for fine-tuning tasks, which include random generation, property optimization and constrained optimization. For all tasks, we follow the setups from previous GNN approaches [20, 52, 33] designing our experiments based on TorchDrug[2] and DIG [29]. We set our disentangled models with a group size of 81, a hessian penalty of 40 and commutativity penalty of 5. In terms of baselines, we include JT-VAE [20], GraphAF [52], GraphDF [33], GCPN [62].

Random generation This task evaluates the quality of randomly generated samples on 4 standard metrics in percentage, including: valid molecules with resampling (Val), valid molecules without resampling (Res), unique molecules (Uni), novel molecules that not appear in the training data (Nov). Following the baselines' setup, we evaluate on ZINC [19], QM9 [46], and MOSES [44]. We train our models for 10 epochs, with a batch size of 32 and the learning rate of 0.001, and compute the metrics over 10K generated molecules. In Table 3 we observe that our CON model achieves the highest score of validity withour resampling on the ZINC dataset, and we note that both UNC and CON are able to improve the performance of GraphAF and GCPN on most metrics, while achieving comparable performance with GraphDF. Since our model is not optimized for molecule generation, the positive results of UNC and CON indicate that disentangled representations may represent way to achieve more expressive generation, and that encouraging the separation of generative factors into different latent codes, may be beneficial for improving the quality of the samples.

Property optimization The goal of the task is to generate novel molecules with high property scores for the penalized logP property, and the quantitative estimation of drug-likeness property (QED). The setup involves pretraining our models for 300 epochs on ZINC random generation, applying the RL fine-tuning procedure from GCPN, and measuring the scores from the top 3 generated molecules. In Table 4 we observe that our models are able to outperform GCPN and GraphAF for both penalized logP and QED, and achieve comparable performance to the more advanced GraphDF. In Figure 3 we show the molecules generated using CON. While our model does not outperform GraphDF, this result is important because it demonstrates that the disentangled representations learned from our models can be leveraged to improve the performance of the downstream task.

[2]https://torchdrug.ai/ (Apache license)

Table 4: Property optimization performance.

	Penalized logP			QED		
	1st	2nd	3rd	1st	2nd	3rd
ZINC	4.52	4.3	4.23	0.948	0.948	0.948
JT-VAE	5.3	4.93	4.49	0.925	0.911	0.910
GCPN	7.98	7.85	7.80	**0.948**	0.947	0.946
GraphAF	12.23	11.29	11.05	**0.948**	**0.948**	0.947
GraphDF	**13.7**	**13.18**	**13.17**	**0.948**	**0.948**	**0.948**
UNC	12.56	12.46	12.03	0.947	**0.948**	0.947
CON	12.89	12.82	12.27	**0.948**	**0.948**	**0.948**

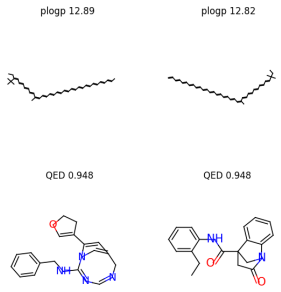

Figure 3: Molecules from CON

Constrained Optimization This task aims to modify the input molecular graph for improving its penalized logP score while keeping the similarity between the input and modified molecules higher than the threshold δ. We follow the setup of GraphAF and GraphDF, which select 800 molecules from ZINC with low penalized logP scores as the input molecules to be optimized. We pretrain our model on the random generation task for 300 epochs, and then fine-tune with the RL procedure from GraphAF. We report the mean and standard deviation of the largest property improvement (Imp), and similarities (Sim) between them and their corresponding input molecules, as well as the success rate (Suc). In Table 5 we compare the results for our CON model with GraphAF and GraphDF. We observe that CON is able to outperform GraphAF 3 out of 4 times for improvement values, and achieves comparable results with GraphDF, which we motivate by the fact that GraphDF provides further fine-tuning in order to improve the baseline. In terms of similarity and success rate, our model has comparable results with the baselines for all δ setups.

Table 5: Constrained optimization performance on 800 molecules used in GraphAF.

δ	GraphAF			GraphDF			CON		
	Imp	Sim	Suc	Imp	Sim	Suc	Imp	Sim	Suc
0.0	$13.13_{\pm 6.89}$	$0.29_{\pm 0.15}$	100	$\mathbf{14.15}_{\pm 6.86}$	$0.29_{\pm 0.13}$	100	$13.26_{\pm 3.24}$	$0.28_{\pm 0.21}$	100
0.2	$11.90_{\pm 6.86}$	$0.33_{\pm 0.12}$	100	$\mathbf{12.77}_{\pm 6.59}$	$0.32_{\pm 0.11}$	100	$12.35_{\pm 5.62}$	$0.32_{\pm 0.13}$	100
0.4	$8.21_{\pm 6.51}$	$0.49_{\pm 0.09}$	99.8	$\mathbf{9.19}_{\pm 6.43}$	$0.48_{\pm 0.08}$	99.6	$8.25_{\pm 4.93}$	$0.50_{\pm 0.18}$	98.4
0.6	$\mathbf{4.98}_{\pm 6.49}$	$0.66_{\pm 0.05}$	96.8	$4.51_{\pm 5.80}$	$0.65_{\pm 0.05}$	92.1	$4.67_{\pm 5.39}$	$0.66_{\pm 0.03}$	93.2

6 Conclusion

We formalize the notion of conditional disentanglement on graphs and propose a novel framework for graph disentanglement by leveraging tools from Lie algebras. Based on the new definition of disentanglement, we design a graph VAE based on a Lie group parameterization, and provide a novel ELBO criteria for optimizing conditional disentanglement. Our method achieves superior performance on quantitative disentanglement benchmarks when compared to contemporary disentangled GNNs and other convolutional layers. Finally, we demonstrate strong capabilities on few-shots classification and molecular generation experiments.

Limitations We rely on synthetic dataset for evaluating disentanglement, because current quantitative metrics are based on the ability to sample each factor and combine them into an observation. Since this is an open problem in representation learning [22], future work will investigate how to provide metrics that can evaluate disentanglement on real-worl d datasets.

References

[1] Inhwan Bae and Hae-Gon Jeon. Disentangled multi-relational graph convolutional network for pedestrian trajectory prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 911–919, 2021.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

[3] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022.

[4] Diane Bouchacourt, Mark Ibrahim, and Stéphane Deny. Addressing the topological defects of disentanglement via distributed operators. *arXiv preprint arXiv:2102.05623*, 2021.

[5] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34 (4):18–42, 2017.

[6] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[7] Hugo Caselles-Dupré, Michael Garcia Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments. *Advances in Neural Information Processing Systems*, 32, 2019.

[8] Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in vaes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2615–2625, 2018.

[9] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *NeurIPS*, 2019.

[10] Kien Do and Truyen Tran. Theory and evaluation metrics for learning disentangled representations. In *International Conference on Learning Representations*, 2019.

[11] Yuanqi Du, Xiaojie Guo, Hengning Cao, Yanfang Ye, and Liang Zhao. Disentangled spatiotemporal graph generative models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[12] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

[13] Marco Fumero, Luca Cosmo, Simone Melzi, and Emanuele Rodolà. Learning disentangled representations via product manifold projection. In *International Conference on Machine Learning*, pages 3530–3540. PMLR, 2021.

[14] Benoit Gaujac, Ilya Feige, and David Barber. Learning disentangled representations with the wasserstein autoencoder. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 69–84. Springer, 2021.

[15] Yunhao Ge, Yunkui Pang, Linwei Li, and Laurent Itti. Graph autoencoder for graph compression and representation learning. In *Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021*, 2021.

[16] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[17] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

[18] Irina Higgins, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P Burgess, Matko Bošnjak, Murray Shanahan, Matthew Botvinick, Demis Hassabis, and Alexander Lerchner. Scan: Learning hierarchical compositional visual concepts. In *International Conference on Learning Representations*, 2018.

[19] John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52 (7):1757–1768, 2012.

[20] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[21] Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR, 2020.

[22] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658, 2018.

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[24] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[25] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[26] Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[27] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 34, 2021.

[28] Xiangyu Li, Zhe Xu, Kun Wei, and Cheng Deng. Generalized zero-shot learning via disentangled representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1966–1974, 2021.

[29] Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Haiyang Yu, Zhao Xu, Jingtun Zhang, Yi Liu, et al. Dig: a turnkey library for diving into graph deep learning research. *Journal of Machine Learning Research*, 22(240):1–9, 2021.

[30] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31, 2018.

[31] Yanbei Liu, Xiao Wang, Shu Wu, and Zhitao Xiao. Independence promoted graph disentangled networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4916–4923, 2020.

[32] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

[33] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, pages 7192–7203. PMLR, 2021.

[34] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *International conference on machine learning*, pages 4212–4221. PMLR, 2019.

[35] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in Neural Information Processing Systems*, 32:2156–2167, 2019.

[36] Graziano Mita, Maurizio Filippone, and Pietro Michiardi. An identifiable double vae for disentangled representations. In *International Conference on Machine Learning*, pages 7769–7779. PMLR, 2021.

[37] Amina Mollaysa, Brooks Paige, and Alexandros Kalousis. Conditional generation of molecules from disentangled representations, 2020. URL `https://openreview.net/forum?id=BkxthxHYvr`.

[38] Milton Llera Montero, Casimir JH Ludwig, Rui Ponte Costa, Gaurav Malhotra, and Jeffrey Bowers. The role of disentanglement in generalisation. In *International Conference on Learning Representations*, 2020.

[39] Milton Llera Montero, Casimir JH Ludwig, Rui Ponte Costa, Gaurav Malhotra, and Jeffrey Bowers. The role of disentanglement in generalisation. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=qbH974jKUVy`.

[40] Francesco Orsini, Paolo Frasconi, and Luc De Raedt. Graph invariant kernels. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[41] Matthew Painter, Adam Prugel-Bennett, and Jonathon Hare. Linear disentangled representations and unsupervised action estimation. *Advances in Neural Information Processing Systems*, 33:13297–13307, 2020.

[42] William Peebles, John Peebles, Jun-Yan Zhu, Alexei Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In *European Conference on Computer Vision*, pages 581–597. Springer, 2020.

[43] David Pfau, Irina Higgins, Alex Botev, and Sébastien Racanière. Disentangling by subspace diffusion. *Advances in Neural Information Processing Systems*, 33:17403–17415, 2020.

[44] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:1931, 2020.

[45] Robin Quessard, Thomas Barrett, and William Clements. Learning disentangled representations and group structure of dynamical environments. *Advances in Neural Information Processing Systems*, 33:19727–19737, 2020.

[46] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[47] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[48] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 287–297. Springer, 2008.

[49] Wulf Rossmann. *Lie groups: an introduction through linear groups*, volume 5. Oxford University Press on Demand, 2006.

[50] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[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(9), 2011.

[52] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2019.

[53] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pages 412–422. Springer, 2018.

[54] Niklas Stoehr, Emine Yilmaz, Marc Brockschmidt, and Jan Stuehmer. Disentangling interpretable generative parameters of random and real-world graphs. *arXiv preprint arXiv:1910.05639*, 2019.

[55] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

[56] Sjoerd Van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? *Advances in Neural Information Processing Systems*, 32, 2019.

[57] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[58] Tan Wang, Zhongqi Yue, Jianqiang Huang, Qianru Sun, and Hanwang Zhang. Self-supervised learning disentangled group representation as feature. *Advances in Neural Information Processing Systems*, 34, 2021.

[59] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 33:20437–20448, 2020.

[60] Tao Yang, Xuanchi Ren, Yuwang Wang, Wenjun Zeng, and Nanning Zheng. Towards building a group-based unsupervised representation disentanglement framework. In *International Conference on Learning Representations*, 2022.

[61] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. *Advances in Neural Information Processing Systems*, 33:20286–20296, 2020.

[62] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

[63] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

[64] Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=bM4Iqfg8M2k`.

[65] Hanlin Zhang, Yi-Fan Zhang, Weiyang Liu, Adrian Weller, Bernhard Schölkopf, and Eric P Xing. Towards principled disentanglement for domain generalization. *arXiv preprint arXiv:2111.13839*, 2021.

[66] Xinqi Zhu, Chang Xu, and Dacheng Tao. Commutative lie group vae for disentanglement learning. In *International Conference on Machine Learning*, pages 12924–12934. PMLR, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Provided in Appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Provided as zip file in the supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specify the experimental setups in Section 5
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See beginning of Section 5
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See beginning of Section 5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.3
   (b) Did you mention the license of the assets? [Yes] See Section 5.3
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our assets in the supplemental material as zip file.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Synthetic datasets

We construct two datasets with the PyTorch Geometric library[3] and the NetworkX library[4], based respectively on Watts-Strogatz, and Erdos-Renyi random graphs. The datasets consist in a list of graphs, where we control the generative factors necessary for synthesizing node feature and adjacency matrices in each graph. This allows us to create labels and latent codes for each graph, similarly to the canonical dataset for disentanglement such as Dsprites[5].

Node feature factors The node feature matrix is computed for both datasets as a random tensor of dimension $N_{\text{nodes}} \times N_{\text{features}}$. The number of node is fixed to 20, while the features are 64. The elements of the feature matrix are sampled from a normal distribution with a given mean and variance, which are two of the generative factors that we control.

[3] `https://github.com/pyg-team/pytorch_geometric`
[4] `https://networkx.org/`
[5] `https://github.com/deepmind/dsprites-dataset`

Adjacency matrix factors The adjacency matrices are computed via the NetworkX library, where we are able to control the factors responsible for generating the graph.

For Watts-Strogatz graphs we have (1) a factor k, which represents the k nearest neighbors in a ring topology, and (2) factor p, which is the probability of rewiring each edge. For Erdos-Renyi graphs, we control only one factor p, that represents the probability for edge creation.

Combination Watts-Strogatz data has 2 adjacency factors, while Erdos-Renyi has only one, thus, in order to achieve the same number of graphs in both datasets, we fix the variance to 2 on Watts-Strogatz, while we provide 20 values in Erdos-Renyi. The final Watts-Strogatz and Erdos-Renyi datasets have $40 \times 40 \times 20 = 32.000$ graphs. The combinations for the factors: mean, variance, k, and p are provided in Table 6. Note that number of nodes is not considered a factor.

Table 6: Graph datasets for disentanglement.

		Node feature		Adjacency matrix		
Dataset	Num nodes	Mean	Variance	k	p	Num graphs
Watts-Strogatz	20	40	1	20	40	32K
Erdos-Renyi	20	40	20	n/a	40	32K

B Unconditional Disentanglement

The unconditional ELBO (Proposition 1), states that given two latent variables Z and T modeling the log-likelihood of the graph data $\mathcal{G}$ is bounded by the following $\mathcal{L}_{\text{unc}}$ ELBO:

$$\mathcal{L}_{\text{unc}} = \mathbb{E}_{q(Z|\mathcal{G})q(T|Z)} \log p(\mathcal{G}|Z)p(Z|T) - \mathbb{E}_{q(Z|\mathcal{G})}\text{KL}(q(T|Z)||p(T)) - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \quad (6)$$

Proof. Using Jensen inequality, we have:

$$\begin{aligned}
\log p(\mathcal{G}) &= \log \int_Z \int_T p(\mathcal{G}, Z, T) \\
&= \log \int_Z \int_T p(\mathcal{G}, Z, T) \frac{q(T|\mathcal{G}, Z)q(Z|\mathcal{G})}{q(T|\mathcal{G}, Z)q(Z|\mathcal{G})} \\
&\geq \int_Z q(Z|\mathcal{G}) \log \int_T p(\mathcal{G}, Z, T) \frac{q(T|\mathcal{G}, Z)}{q(T|\mathcal{G}, Z)q(Z|\mathcal{G})} \\
&= \int_Z q(Z|\mathcal{G}) \log \int_T p(\mathcal{G}, Z, T) \frac{q(T|\mathcal{G}, Z)}{q(T|\mathcal{G}, Z)} - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G})
\end{aligned}$$

By applying Jensen inequality again, we obtain:

$$\begin{aligned}
\log p(\mathcal{G}) &\geq \int_Z q(Z|\mathcal{G}) \int_T q(T|\mathcal{G}, Z) \log \frac{p(\mathcal{G}, Z, T)}{q(T|\mathcal{G}, Z)} - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \\
&= \int_Z q(Z|\mathcal{G}) \int_T q(T|\mathcal{G}, Z) \log \frac{p(\mathcal{G}, Z|T)p(T)}{q(T|\mathcal{G}, Z)} - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \\
&= \mathbb{E}_{q(Z|\mathcal{G})}\mathbb{E}_{q(T|\mathcal{G},Z)} \log p(\mathcal{G}, Z|T) - \mathbb{E}_{q(Z|\mathcal{G})} \int_T q(T|\mathcal{G}, Z) \log \frac{q(T|\mathcal{G}, Z)}{p(T)} - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \\
&= \mathbb{E}_{q(Z|\mathcal{G})}\mathbb{E}_{q(T|\mathcal{G},Z)} \log p(\mathcal{G}, Z|T) - \mathbb{E}_{q(Z|\mathcal{G})}\text{KL}(q(T|\mathcal{G}, Z)||p(T)) - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G})
\end{aligned}$$

Assuming Markov settings, which imply $q(T|Z) = q(T|\mathcal{G}, Z)$ and $p(\mathcal{G}|Z, T) = p(\mathcal{G}|Z)$, we have:

$$\log p(\mathcal{G}) = \mathbb{E}_{q(Z|\mathcal{G})q(T|Z)} \log p(\mathcal{G}|Z)p(Z|T) - \mathbb{E}_{q(Z|\mathcal{G})}\text{KL}(q(T|Z)||p(T)) - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G})$$

□

C Conditional Disentanglement

Proof. Using Jensen's inequality, we note that

$$\begin{aligned}
\log P(\mathcal{G}) &= \log \int_Z \int_T \int_{\hat{Z}} p(\mathcal{G}, Z, T, \hat{Z}) \\
&= \log \int_Z \int_T \int_{\hat{Z}} p(\mathcal{G}, Z, T, \hat{Z}) \frac{q(Z,T|\mathcal{G})}{q(Z,T|\mathcal{G})} \\
&= \log \int_Z \int_T \int_{\hat{Z}} p(\mathcal{G}, Z, T, \hat{Z}) \frac{q(T|\mathcal{G},Z)q(Z|\mathcal{G})}{q(T|\mathcal{G},Z)q(Z|\mathcal{G})} \\
&\geq \int_Z q(Z|\mathcal{G}) \log \int_T \int_{\hat{Z}} p(\mathcal{G}, Z, T, \hat{Z}) \frac{q(T|\mathcal{G},Z)}{q(T|\mathcal{G},Z)q(Z|\mathcal{G})} \\
&= \int_Z q(Z|\mathcal{G}) \log \int_T \int_{\hat{Z}} p(\mathcal{G}, Z, T, \hat{Z}) \frac{q(T|\mathcal{G},Z)}{q(T|\mathcal{G},Z)} - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G})
\end{aligned}$$

Using our conditional independence assumption

$$q(T|\mathcal{G}, Z) = q(T|Z) \ ,$$

and applying Jensen's inequality again, we get

$$\begin{aligned}
\log P(\mathcal{G}) &\geq \int_Z q(Z|\mathcal{G}) \log \int_T \int_{\hat{Z}} p(\mathcal{G}, Z, T, \hat{Z}) \frac{q(T|Z)}{q(T|Z)} - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \\
&\geq \int_Z q(Z|\mathcal{G}) \int_T q(T|Z) \log \int_{\hat{Z}} \frac{p(\mathcal{G}, Z, T, \hat{Z})}{q(T|Z)} - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \\
&= \mathbb{E}_{q(Z|\mathcal{G})} \mathbb{E}_{q(T|Z)} \log \int_{\hat{Z}} \frac{p(\mathcal{G}, Z, T, \hat{Z})}{q(T|Z)} - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \\
&= \mathbb{E}_{q(Z|\mathcal{G})} \mathbb{E}_{q(T|Z)} \log \frac{p(T)p(\hat{Z}|T)p(Z|\hat{Z},T)p(\mathcal{G}|\hat{Z},T,Z)}{q(T|Z)} - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \\
&= \mathbb{E}_{q(Z|\mathcal{G})} \mathbb{E}_{q(T|Z)} \left(\log p(Z|\hat{Z},T)p(\mathcal{G}|\hat{Z},T,Z) + \log \frac{p(T)p(\hat{Z}|T)}{q(T|Z)} \right) - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G})
\end{aligned}$$

Finally, assuming

$$\mathbb{E}_{q(Z|\mathcal{G})} p(\mathcal{G}|\hat{Z}, Z) = \mathbb{E}_{q(Z|\mathcal{G})} \mathbb{E}_{q(T|Z)} p(Z|\hat{Z}, T) p(\mathcal{G}|\hat{Z}, T, Z) \ ,$$

we get

$$\begin{aligned}
\log P(\mathcal{G}) &\geq \mathbb{E}_{q(Z|\mathcal{G})} \mathbb{E}_{q(T|Z)} \left(\log p(\mathcal{G}|\hat{Z}, Z) + \log \frac{p(T)p(\hat{Z}|T)}{q(T|Z)} \right) - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \\
&= \mathbb{E}_{q(Z|\mathcal{G})q(T|Z)} \log p(\mathcal{G}|\hat{Z}, Z) p(\hat{Z}|T) - \mathbb{E}_{q(Z|\mathcal{G})} \mathrm{KL}(q(T|Z)||p(T)) - \mathbb{E}_{q(Z|\mathcal{G})} \log q(Z|\mathcal{G}) \ .
\end{aligned}$$

□

D Scalability experiment

We perform a graph classification task on large scale datasets, and report their dimensions in Table 7. In Table 8 we report the results for our CON model on social network datasets including IMDB binary IMDB-multi, COLLAB, as well as macro molecules such as PROTEINS and MUTAG. The baselines for this experiment include FactorGCN [61] and DGCL [27]. For our model we set group-size to 81, hessian penalty to 40 and commutative penalty to 5. We follow the classification setup from Yang et al. [61], with a ten-fold cross-validation procedure, and report accuracy and standard deviation. The results show that both UNC and CON are able to improve the performance of the baselines. In particular, CON achieves the highest accuracy on all the datasets.

Table 7: Dimensions of datasets

	IMDB-B	IMDB-M	COLLAB	PROTEINS	MUTAG
Graphs	1000	500	5000	1113	188
Classes	2	3	3	2	2
Avg. Nodes	19.77	13.00	74.49	39.06	17.93
Avg. Edges	96.53	65.94	2457.78	72.82	19.79

Table 8: Scalability

	IMDB-B	IMDB-M	COLLAB	PROTEINS	MUTAG
FactorGCN	75.3 ± 2.7	-	81.2 ± 1.4	-	89.9 ± 6.5
DGCL	75.9 ± 0.7	51.9 ± 0.4	81.2 ± 0.3	76.4 ± 0.5	92.1 ± 0.8
UNC	76.1 ± 0.2	51.2 ± 0.9	81.9 ± 0.8	76.5 ± 1.5	91.4 ± 1.2
CON	**76.5 ± 0.3**	**52.5 ± 0.2**	**82.5±0.2**	**76.9 ± 0.1**	**92.5 ± 0.5**

E Correlation analysis

Following previous work [61, 34], we report the correlation matrix of the hidden graph features. The correlation matrix aims to show whether the features capture mutually exclusive information, which is achieved when the there is a block-wise correlation pattern. Specifically, we follow the setup from Yang et al. [61], which analyse the correlation matrix on a dataset composed of graphs, where the generative factors are synthetic factor graphs.

In our models we use a group-size of 32 (to match with hidden features size of the other methods), a hessian penalty of 40 and commutative penalty of 5. In Figure 4 we note that both our UNC and CON models achieve a block-wise correlation pattern, the conditional version has also some other highlighted regions, capturing the entangled factors.

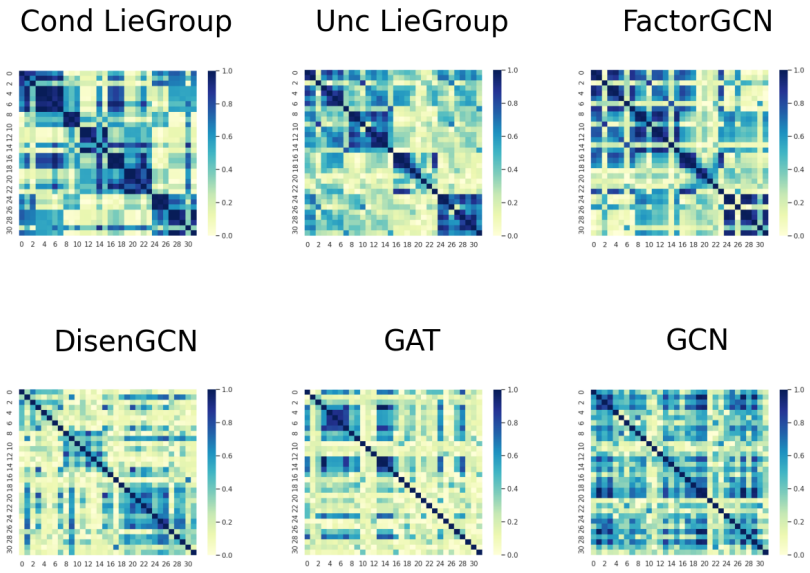

Figure 4: Correlation analysis

F Learned graph variations

In this experiment we train the CON model on Watts-Strogatz (WS) random graphs, which have 3 generative factors, n for number of nodes, p for connectivity (the probability of rewiring each edge), and k for neighborhood (k nearest neighbors in a ring topology). We then follow Stoehr et al. [54] and project latent variables in pairs to see how the factors vary.

In Figure 5 we report the traversal of the CON model trained on 3 latent variables, and we observe the following.

- The variable z_0 controls the node attribute value which is indicated by the color in the right column. For example, in the top row on the right, we see the color changing from blue to white along the horizontal axis.
- The variable z_1 controls connectivity, for example on top row, following the y axis, we see that the edges decrease, and in second row, along the x axis, edges increase.
- The variable z_2 controls the number of nodes, and we see that in the middle row and the last row the number of nodes decreases along the vertical axis.

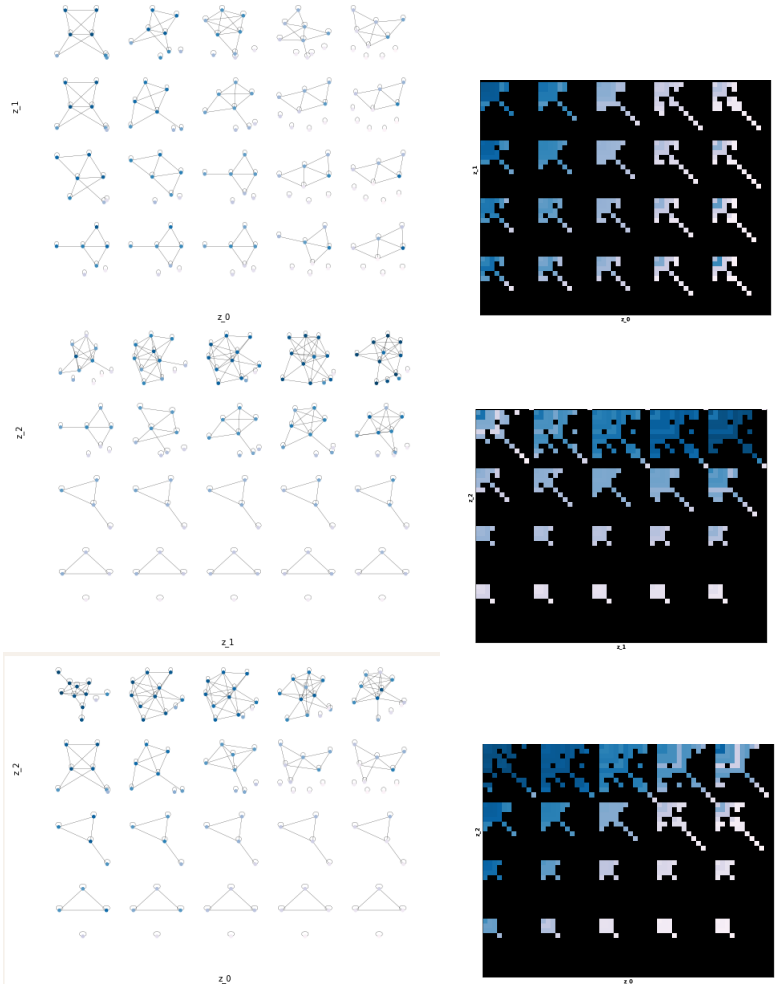

Figure 5: Left: Graph traversals. Right: traversals of node attribute values. Top row: z_0, z_1, middle row: z_1, z_2, bottom row: z_0, z_2.

G Molecular experiment

In order to have a fair comparison with Flow models, such as GraphAF [52] and GraphDF [33], we incorporate our parameterization into the training process of a Flow model. Both GraphAF and GraphDF are designed using relational GCN [50] (RGCN) as building block to compute an embedding of a graph using a layer R-GCN: $H_i^L = \text{R-GCN}(G_i)$, $\tilde{h}_i = \text{sum}(H_i^L)$ where sum denotes the sum-pooling operation, and $H_{i,j}^L \in \mathbb{R}^k$ denotes the embedding of the j-th node in the embeddings H_i^L.

We incorporate some parts of our models after the RGCN encoding, as follows. From the embedding H, we compute the lie algebra coordinates, with an encoder $q(T|H)$, following our E_{group} network from Section 3. We then apply the lie algebra exponential mapping to reconstruct the embedding H as $p(H|T)$, following the D_{group} network from Section 3. Furthermore, we incorporate the hessian penalty as a regularization. In all the three tasks, we follow the experimental setup from GraphAF, we set the model with group-size of 81, and a hessian penalty of 40.

We first perform the the random graph generation task, which involves quantifying 4 metrics in percentage, including: valid molecules with resampling (Val), valid molecules without resampling (Res), unique molecules (Uni), and novel molecules not appearing in the training data (Nov). In Table 9 we report results for GCPN, GraphAF, GraphDF and our models. The results indicate that with the inclusion of a lie algebra reparameterization and the hessian penalty, the performance on random generation is enhanced, and the Flow + CON model achieves the best performance on all metrics a part from uniqueness, where it achieves comparable results.

Table 9: Random graph generation.

	ZINC				QM9				MOSES			
Model	Val	Res	Uni	Nov	Val	Res	Uni	Nov	Val	Res	Uni	Nov
GCPN	**100**	20	**99.97**	**100**	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
GraphAF	**100**	68	99.1	**100**	**100**	67	94.15	88.83	**100**	71	**99.99**	**100**
GraphDF	**100**	89.03	99.16	**100**	**100**	82.67	**97.62**	98.1	**100**	87.58	99.55	**100**
UNC	**100**	78.21	99.12	**100**	**100**	75.43	95.22	90.33	**100**	81.62	99.62	**100**
CON	**100**	86.02	99.14	**100**	**100**	82.92	96.43	92.28	**100**	84.29	99.79	**100**
Flow + CON	**100**	**90.12**	99.53	**100**	**100**	**83.45**	97.46	**98.62**	**100**	**88.42**	99.79	**100**

Secondly, we report the results for the property optimization task in Table 10, which show the scores from the top 3 generated molecules for two selected properties (penalized logp and QED). The baselines are the same ones as in the random generation task. We observe that our Flow + CON model is the top performing method on penalized logP for the top 3 scores, and it achieves the top results, together with the other methods, for QED.

Table 10: Property optimization performance.

	Penalized logP			QED		
	1st	2nd	3rd	1st	2nd	3rd
ZINC	4.52	4.3	4.23	0.948	0.948	0.948
GCPN	7.98	7.85	7.80	**0.948**	0.947	0.946
GraphAF	12.23	11.29	11.05	**0.948**	**0.948**	0.947
GraphDF	13.7	13.18	13.17	**0.948**	**0.948**	**0.948**
UNC	12.56	12.46	12.03	0.947	**0.948**	0.947
CON	12.89	12.82	12.27	**0.948**	**0.948**	**0.948**
Flow + CON	**13.97**	**13.35**	**13.39**	**0.948**	**0.948**	**0.948**

Finally, we show in Table 11 the results for the constrained optimization task. We report only GraphDF as baseline, which is the top performing model among the baselines. The evaluation involves reporting the mean and standard deviation of metrics including: the largest property improvement (Imp), and similarities (Sim) between them and their corresponding input molecules, as well as the success rate (Suc).

We observe that by combining the disentangled principles developed in our models into a Flow-based model, we are able to further improve the results, and achieve the top performance.

Table 11: Constrained optimization performance on 800 molecules used in GraphAF.

δ	Flow + CON			GraphDF			CON		
	Imp	Sim	Suc	Imp	Sim	Suc	Imp	Sim	Suc
0.0	**14.28**$_{\pm 5.24}$	0.29$_{\pm 0.10}$	100	14.15$_{\pm 6.86}$	0.29$_{\pm 0.13}$	100	13.26$_{\pm 3.24}$	0.28$_{\pm 0.21}$	100
0.2	**12.89**$_{\pm 4.33}$	0.34$_{\pm 0.09}$	99.93	12.77$_{\pm 6.59}$	0.32$_{\pm 0.11}$	100	12.35$_{\pm 5.62}$	0.32$_{\pm 0.13}$	100
0.4	**9.32**$_{\pm 5.45}$	0.50$_{\pm 0.03}$	99.3	9.19$_{\pm 6.43}$	0.48$_{\pm 0.08}$	99.6	8.25$_{\pm 4.93}$	0.50$_{\pm 0.18}$	98.4
0.6	**4.98**$_{\pm 6.21}$	0.67$_{\pm 0.04}$	97.3	4.51$_{\pm 5.80}$	0.65$_{\pm 0.05}$	92.1	4.67$_{\pm 5.39}$	0.66$_{\pm 0.03}$	93.2