Interpretable Geometric Deep Learning via Learnable Randomness Injection

Anonymous Author(s)
Affiliation
Address
email

Abstract

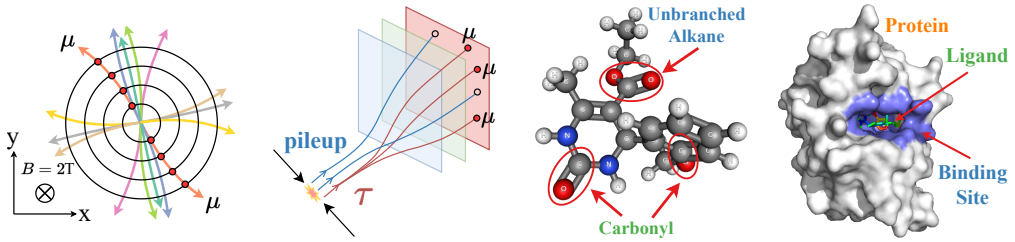
Point cloud data is ubiquitous in scientific fields. Recently, geometric deep learning (GDL) has been widely applied to solve prediction tasks with such data. However, GDL models are often complicated and hardly interpretable, which poses concerns to scientists when deploying these models in scientific analysis and experiments. This work proposes a general mechanism named *learnable randomness injection* (LRI), which allows building inherently interpretable models based on general GDL backbones. Once being trained, LRI-induced models can detect the points in the point cloud data that carry information indicative to the prediction label. Such indicative information may be reflected by either the existence of these points in the data or the geometric locations of these points. We also propose four datasets from real scientific applications in the domains of high energy physics and biochemistry to evaluate LRI. Compared with previous post-hoc interpretation methods, the points detected by LRI align much better and stabler with the ground-truth patterns that have actual scientific meanings. LRI-induced models are also more robust to the distribution shifts between training and test scenarios.

1 Introduction

The measurement of many scientific research objects can be represented as a point cloud, i.e., a set of featured points in some geometric space. For example, in high energy physics (HEP), particles generated from collision experiments leave spacial signals on the detectors they pass through [1]; In biology, a protein is often measured and represented as a collection of amino acids with spacial locations [2, 3]. Geometric quantities of such point cloud data often encode important properties of the research object, analyzing which researchers may expect new scientific discoveries [4, 5].

Recently, machine learning techniques have been employed to accelerate the procedure of scientific discovery [6, 7]. For geometric data as above, geometric deep learning (GDL) [8, 9] has shown great promise and has been applied to the fields such as HEP [10, 11], biochemistry [12, 13] and so on. However, geometric data in practice is often irregular and high-dimensional. Think about a collision event in HEP that generates hundreds to thousands of particles, or a protein that consists of tens to hundreds of amino acids. Although each particle or each amino acid is located in a low-dimensional space, the whole set of points eventually is extremely irregular and high-dimensional. So, current research on GDL primarily focuses on designing neural network (NN) architectures for GDL models to deal with the above data challenge. GDL models have to preserve the symmetries of the system and incorporate the inductive biases reflected by geometric principles to guarantee their prediction quality [14, 15], and therefore often involve dedicated-designed complex NN architectures.

Submitted to NeurIPS 2021 AI for Science Workshop.



(a) ActsTrack (b) Tau3mu (c) SynMol (d) PLBind

Figure 1: Illustrations of the four scientific datasets in this work to study interpretable GDL models.

Albeit with outstanding prediction performance, the complication behind GDL models makes them hardly interpretable. However, in many scientific applications, interpretable models are in need [16]: For example, in drug discovery, compared with just predicting the binding affinity of a protein-ligand pair, it is more useful to know which groups of amino acids determine the affinity and where can be the binding site, as the obtained knowledge may guide future research directions [17–19]. Moreover, scientists tend to only trust interpretable models in many scenarios, e.g., most applications in HEP, where data from real experiments lack labels and models have to be trained on simulation data [20]. Here, model interpretation is used to verify if a model indeed captures the patterns that match scientific principles instead of some spurious correlation between the simulation environment and labels. Unfortunately, to the best of our knowledge, there have been no studies on interpretable GDL models let alone their applications in scientific problems. Some previous post-hoc methods may be extended to interpret a pre-trained GDL model while they suffer from some limitations as to be reviewed in Appendix A. Moreover, recent works [21–24] have shown that the data patterns detected by post-hoc methods are often inconsistent across interpretation methods and pre-trained models, and may hardly offer reliable scientific insights.

To fill the gap, this work proposes to study interpretable GDL models. Inspired by the recent work [24], we first propose a general mechanism named *Learnable Randomness Injection* (LRI) that allows building inherently interpretable GDL models based on a broad range of GDL backbones. We then propose four datasets from real-world scientific applications in HEP and biochemistry and provide an extensive comparison between LRI-induced GDL models and previous post-hoc interpretation approaches (after being adapted to GDL models) over these datasets.

Our LRI mechanism provides model interpretation by detecting a subset of points from the point cloud that is most likely to determine the label of interest. The idea of LRI is to inject learnable randomness to each point, where, along with training the model for label prediction, injected randomness on the points that are important to prediction gets reduced. The convergent amounts of randomness on points essentially reveal the importance of the corresponding points for prediction. Specifically in GDL, as the importance of a point may be indicated by either the existence of this point in the system or its geometric location, we propose to inject two types of randomness, *Bernoulli randomness*, with the framework name *LRI-Bernoulli* to test *existence importance* of points and *Gaussian randomness* on geometric features, with the framework name *LRI-Gaussian* to test *location importance* of points. Moreover, by properly parameterized such Gaussian randomness, we may tell for a point, how in different directions perturbing its location affects the prediction result more. With such fine-grained geometric information, we may estimate the direction of the particle velocity when analyzing particle collision data in HEP. LRI is theoretically sound as it essentially uses a variational objective derived from the information bottleneck principle [25]. LRI-induced models also show better robustness to the distribution shifts between training and test scenarios, which gives scientists more confidence in applying them in practice.

We note that one obstacle to studying interpretable GDL models is the lack of valid datasets that consist of both classification labels and scientifically meaningful patterns to verify the quality of interpretation. Therefore, another significant contribution of this work is to prepare four benchmark datasets grounded on real-world scientific applications to facilitate interpretable GDL research. These datasets cover important applications in HEP and biochemistry. We briefly illustrate these four datasets in Fig. 1 and briefly introduce them in the following.

- `ActsTrack` is a particle tracking dataset in HEP that is used to reconstruct the properties, such as the kinematics of a charged particle given a set of position measurements from a tracking detector. Tracking is an indispensable step in analyzing HEP experimental data as well as particle tracking used in medical applications such as proton therapy [26–28]. *Our task* is formulated differently from traditional track reconstruction tasks: We predict the existence of a $z \to \mu\mu$ decay and use the set of points from the μ's to verify model interpretation, which can be used to reconstruct μ tracks. `ActsTrack` also provides a controllable environment (e.g., magnetic field strength) to study fine-grained geometric patterns.
- `Tau3Mu`, another application in HEP, is to detect a challenging signature of charged lepton flavor violating decays, i.e., the $\tau \to \mu\mu\mu$ decay, given simulated muon detector hits in proton-proton collisions. Such decays are greatly suppressed in the Standard Model (SM) of particle physics [29, 30], therefore, any detection of them is a clear signal of new physics beyond the Standard Model [31, 32]. Unfortunately, $\tau \to \mu\mu\mu$ contains particles of extremely low momentum, thus technologically impossible to trigger with traditional human-engineered algorithms. Hence, online detection with advanced models is required to capture such decays at the Large Hadron Collider. *Our task* is to predict the existence of $\tau \to \mu\mu\mu$ and use the detector hits left by the μ's to verify model interpretation.
- `SynMol` is a molecular property prediction task. Although some works have studied model interpretability in such tasks [33, 34], they limit their focus on the chemical-bond-graph representations of molecules, and largely ignore their geometric features. In this work, we put focus on 3D molecular representations. *Our task* is to predict the property given by two functional groups carbonyl and unbranched alkane [33] and use atoms in them to verify model interpretation.
- `PLBind` is to predict protein-ligand binding affinities given the 3D structures of proteins and ligands, which is a crucial step in drug discovery, because a high affinity is one of the major drug selecting criteria [35, 18]. Accurately predicting their affinities with interpretable models is useful for rational drug design and may help the understanding of the underlying biophysical mechanism that enables binding [36–38]. *Our task* is to predict if the affinity is above a threshold and use amino acids in the binding site of the test protein to verify model interpretation.

We evaluate LRI with three popular GDL backbone models over the above datasets. We also extend five baseline interpretation methods to GDL for comparison. We find that interpretation results given by LRI align much better with the scientific facts than those of the baselines. Also, we observe over some datasets, LRI-Gaussian outperforms LRI-Bernoulli while on others vice versa. This implies different GDL applications may have different interpretation requirements. Information indicative of the labels may vary regarding how it depends on the geometric features of the points. Interestingly, we find LRI-Gaussian can discover some fine-grained geometric patterns, such as providing high-quality estimations of the directions of particle velocities in `ActsTrack`, and a high-quality estimation of the strength of the used magnetic field. Moreover, neither of LRI mechanisms degrades the prediction performance of the used backbone models. LRI mechanisms even improve model generalization when there exist some distribution shifts between the training and test scenarios.

2 Preliminaries and Problem Formulation

In this section, we define some useful concepts and notations, while detailed review of related works can be found in Appendix A.

GDL Tasks. We consider a data sample is a point cloud $\mathcal{C} = (\mathcal{V}, \mathbf{X}, \mathbf{r})$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is a set of n points, $\mathbf{X} \in \mathbb{R}^{n \times d}$ includes d-dimensional features for all points, and $\mathbf{r} \in \mathbb{R}^{n \times 3}$ denotes 3D spacial coordinates of points. In this work, we introduce our notations by assuming the points are in 3D euclidean space while our method can be generalized. We focus on building a *classification* model $\hat{y} = f(\mathcal{C})$ to predict the class label y of $\mathcal{C}$. Regression tasks are left for future studies.

GDL Models. The *first* class of DGL models view each sample of points as an unordered set. It learns a dense representation $\mathbf{z}_v$ for each $v \in \mathcal{V}$, and then applies a permutation invariant function, e.g., sum/mean/max pooling, to aggregate all point representations so that they can handle irregular

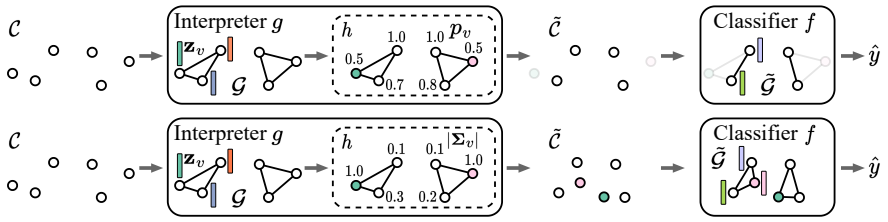


Figure 2: The architectures of LRI-Bernoulli (top) and LRI-Gaussian (bottom).

data [39, 40]. The *second* class of methods can better utilize geometric features and local information. These methods first construct a k-nn graph $\mathcal{G}$ over the points in each sample based on their distances, e.g., $\|\mathbf{r}_v - \mathbf{r}_u\|$, and iteratively update the representation of point v via aggregation $\text{AGG}(\{\mathbf{z}_u \mid u \in \mathcal{N}(v)\})$, where $\mathcal{N}(v)$ is the neighbours of point v in graph $\mathcal{G}$ and AGG is a permutation invariant function. Then, another function is used to aggregate all point representations to make predictions. Compared with graph neural networks (GNNs) [41–43] that encode graph-structured data without geometric features, GDL models often process geometric features carefully: Typically, these features are transformed into some scalars such as distances, angles and used as features so that some group (e.g., $E(3), SO(3)$) invariances of the prediction can be kept [44–48]. Some models that perform 3D convolution over 3D data also belong to the second class because the convolution kernels can be viewed as one way to define the distance scalars for graph construction and neighborhood aggregation [49–51]. The *third* class will dynamically construct the k-nn graphs based on the hidden representations of points [52, 11, 53]. In this work, we focus on using the second class of models' architectures as the backbones because most scientific applications adopt this class of models.

Interpretable Patterns in GDL. Given a sample $\mathcal{C} = (\mathcal{V}, \mathbf{X}, \mathbf{r})$, our goal of building an inherently interpretable model is that the model by itself can identify a subset of points $\mathcal{C}_s = (\mathcal{V}_s, \mathbf{X}_s, \mathbf{r}_s)$ that best indicates the label y. Mostly, $\mathcal{C}_s$ will have a scientific meaning. For example, in the task to detect $\tau \to \mu\mu\mu$ decay, our model should identify the detector hits left by the three μ's but not the hits from other particles. We consider two types of indications of the label given by a point: *existence importance*, i.e., whether the point exists in the cloud is important to determine the label, and *location importance*, i.e. whether the geometric location of the point is important to determine the label. In the above example, the existence of the detector hits left by the three μ's is of course important. On the other hand, the locations of these hits are also crucial because location features reflect the momentum of these particles when they pass through detectors, which should satisfy equations regarding certain invariant mass if they are indeed generated from a $\tau \to \mu\mu\mu$ decay.

3 Methodology

In this section, we introduce our method *Learnable Randomness Injection* (LRI). In the high-level framework of LRI, we have an interpreter g and a classifier f. g is used to encode the original data and generate randomness to perturb the data. f is used to encode the perturbed data and make predictions. g and f are trained together to make accurate predictions while providing interpretability. LRI can be applied to a large class of GDL models to make them interpretable. We may choose a GDL model architecture as the backbone to build the data encoders in g and f. These encoders could share or not share parameters. Below, we will introduce specifics about this procedure. We first describe LRI-Bernoulli, where Bernoulli randomness is injected to measure the *existence important* of points. Finally, we introduce LRI-Gaussian, which injects Gaussian randomness into geometric features to test the *location importance* of points. We also show in Appendix B that our objectives are essentially the variational upper bounds of the information bottleneck principle [25], which makes LRI theoretically sound.

3.1 LRI-Bernoulli to Test Existence Importance

Pipeline. Given a sample $\mathcal{C} = (\mathcal{V}, \mathbf{X}, \mathbf{r})$, we first construct a k-nn graph $\mathcal{G}$ based on the euclidean distance $\|\mathbf{r}_v - \mathbf{r}_u\|$ between every pair of points $v, u \in \mathcal{V}$. As shown in the top of Fig. 2, the

interpreter g encodes $\mathcal{C}$, generates a representation $\mathbf{z}_v$ for each point v and uses the last component h to map $\mathbf{z}_v$ to $p_v \in [0, 1]$. Here, h consists of an MLP plus a sigmoid layer, and samples a Bernoulli mask for each point via $m_v \sim \text{Bern}(p_v)$. The sampling is based on a reparameterization trick [54, 55] to make $\frac{dm_v}{dp_v}$ computable. The perturbed data $\tilde{\mathcal{C}}$ is yielded by removing the points with $m_v = 0$ in $\mathcal{C}$. The edges in $\mathcal{G}$ connected to these points are also masked and removed, which gives a graph $\tilde{\mathcal{G}}$. Finally, the classifier f takes as inputs $\tilde{\mathcal{C}}$ and $\tilde{\mathcal{G}}$ to make predictions.

Objective. Eq. 1 shows our objective for each sample $\mathcal{C}$, where the first term is a cross-entropy loss for classification and the second term is a KL divergence regularizer. β is the regularization coefficient and $\text{Bern}(\alpha)$ is a predefined Bernoulli distribution with hyperparameter $\alpha < 1$.

$$\min \mathcal{L}_{CE}(f(\tilde{\mathcal{C}}, \tilde{\mathcal{G}}), y) + \beta \sum_{v \in \mathcal{V}} D_{\text{KL}}(\text{Bern}(p_v) \| \text{Bern}(\alpha)). \tag{1}$$

Here, f is optimized via the first term. The interpreter g is optimized via the gradients that pass through the masks $\{m_v\}_{v \in \mathcal{V}}$ contained in $\tilde{\mathcal{C}}$ and $\tilde{\mathcal{G}}$ in the first term, and $\{p_v\}_{v \in \mathcal{V}}$ in the second term.

Interpretation Rationale. The interpretation is given by the competition between the two terms in Eq. 1. The first term is to achieve good classification performance so it tends to denoise the data $\tilde{\mathcal{C}}$ by reducing the randomness generated by g, i.e., $p_v \to 1$. The second term, on the other hand, tends to keep the level of randomness, i.e., $p_v \to \alpha$. After training, p_v will converge to some value. If the existence of a point v is important to the label y, then p_v should be close to 1. Otherwise, p_v is close to α. We use p_v's to rank the points $v \in \mathcal{V}$ according to their existence importance.

3.2 LRI-Gaussian to Test Location Importance

Pipeline. We start from the same graph $\mathcal{G}$ as LRI-Bernoulli. As shown in the bottom of Fig. 2, here, the interpreter g will encode the data and map it to a covariance matrix $\mathbf{\Sigma}_v \in \mathbb{R}^{3\times3}$ for each point v. Gaussian randomness $\epsilon_v \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_v)$ is then sampled to perturb the geometric features $\tilde{\mathbf{r}}_v = \mathbf{r}_v + \epsilon_v$ of v. Note that, to test location importance, a new k-nn graph $\tilde{\mathcal{G}}$ is constructed based on perturbed distances $\|\tilde{\mathbf{r}}_v - \tilde{\mathbf{r}}_u\|$. Reconstructing $\tilde{\mathcal{G}}$ is necessary because using the original graph $\mathcal{G}$ will leak information from the original geometric features $\mathbf{r}$. Finally, the classifier f takes as inputs the location-perturbed data $\tilde{\mathcal{C}} = (\mathcal{V}, \mathbf{X}, \tilde{\mathbf{r}})$ and $\tilde{\mathcal{G}}$ to make predictions.

Objective. Eq. 2 shows the objective of LRI-Gaussian for each sample $\mathcal{C}$. Different from Eq. 1, here the regularization is given by a predefined Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma\mathbf{I})$, where $\mathbf{I}$ is an identity matrix and σ is a hyperparameter.

$$\min \mathcal{L}_{CE}(f(\tilde{\mathcal{C}}, \tilde{\mathcal{G}}), y) + \beta \sum_{v \in \mathcal{V}} D_{\text{KL}}(\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_v) \| \mathcal{N}(\mathbf{0}, \sigma\mathbf{I})). \tag{2}$$

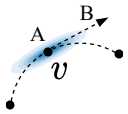
Again, the classifier f will be optimized via the first term. The interpreter g will be optimized via the gradients that pass through the perturbation $\{\epsilon_v\}_{v \in \mathcal{V}}$ implicitly contained in $\tilde{\mathcal{C}}$ and $\tilde{\mathcal{G}}$ in the first term, and $\{\mathbf{\Sigma}_v\}_{v \in \mathcal{V}}$ in the second term. However, there are two technical difficulties to be addressed.

First, how to parameterize $\mathbf{\Sigma}_v$ as it should be positive definite, and then how to make $\frac{d\epsilon_v}{d\mathbf{\Sigma}_v}$ computable? Our solution is to let the last component h in g map representation $\mathbf{z}_v$ not to $\mathbf{\Sigma}_v$ directly but to a dense matrix $\mathbf{U}_v \in \mathbb{R}^{3\times3}$ via an MLP and two scalars $a_1, a_2 \in \mathbb{R}^+$ via a softplus layer. Then, the covariance matrix is computed by $\mathbf{\Sigma}_v = a_1\mathbf{U}_v\mathbf{U}_v^T + a_2\mathbf{I}$. Moreover, we find using $\mathbf{\Sigma}_v$ and the reparameterization trick for multivariate Gaussian implemented by PyTorch is numerically unstable as it includes Cholesky decomposition. So, instead, we use the reparameterization trick $\epsilon_v = \sqrt{a_1}\mathbf{U}_v\mathbf{s}_1 + \sqrt{a_2}\mathbf{I}\mathbf{s}_2$, where $\mathbf{s}_1, \mathbf{s}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. It is not hard to show that $\mathbb{E}[\epsilon_v\epsilon_v^T] = \mathbf{\Sigma}_v$.

Second, the construction of the k-nn graph $\tilde{\mathcal{G}}$ based on $\tilde{\mathbf{r}}_v$ is not differentiable, which makes the gradients of $\{\epsilon_v\}_{v \in \mathcal{V}}$ that pass through the structure of $\tilde{\mathcal{G}}$ not computable. We address this issue by associating each edge v, u in $\tilde{\mathcal{G}}$ with a weight $w_{vu} \in (0, 1)$ that monotonically decreases w.r.t. the distance, $w_{vu} = \phi(\|\tilde{\mathbf{r}}_v - \tilde{\mathbf{r}}_u\|)$. These weights are used in the neighborhood aggregation procedure in f. Specifically, for the central point v, f adopts aggregation $\text{AGG}(\{w_{vu}\mathbf{z}_u \mid u \in \mathcal{N}(v)\})$, where $\mathbf{z}_u$ is the representation of the neighbor point u in the current layer. This design makes the structure

of $\tilde{\mathcal{G}}$ differentiable. Moreover, because we set $w_{uv} < 1$, the number of used nearest neighbors to construct $\tilde{\mathcal{G}}$ is "conceptually" smaller than k. So, in practice, we choose a slightly larger number (say $1.5k$) of nearest neighbors to construct $\tilde{\mathcal{G}}$ and adopt the above strategy.

Interpretation Rationale. The interpretation rationale is similar to that of LRI-Bernoulli, i.e., given by the competition between the two terms in Eq. 1. The first term is to achieve good classification performance by reducing the randomness generated by g. The second term, on the other hand, tends to keep the level of randomness, i.e., $\mathbf{\Sigma}_v \to \sigma\mathbf{I}$. After training, the convergent determinant $|\mathbf{\Sigma}_v|$ which characterizes the entropy of injected Gaussian randomness, indicates the *location importance* of point v. We use $|\mathbf{\Sigma}_v|$'s to rank the points $v \in \mathcal{V}$ according to their location importance.



Fine-grained Interpretation on Location Importance. Interestingly, the convergent $\mathbf{\Sigma}_v$ implies more fine-grained geometric information, i.e., how different directions of perturbations on point v affect the prediction. This can be analyzed by checking the eigenvectors of $\mathbf{\Sigma}_v$. As illustrated in the figure on the right, $\mathbf{\Sigma}_v$ of point v at A is represented by the ellipses $\{\mathbf{x} : \mathbf{x}^T\mathbf{\Sigma}_v\mathbf{x} < \theta\}$ for different θ's. It tells perturbing v towards the direction B affects the prediction less than perturbing v towards the orthogonal direction. As a showcase, later, we use such fine-grained information to conduct an in-depth analysis of HEP data.

4 Benchmarking Interpretable GDL

In this section, we will evaluate LRI-Bernoulli, LRI-Gaussian and some baseline methods extended to GDL over the proposed four datasets. Here, we briefly describe our setup and put more details on the datasets, hyperparameter tuning, and method implementations, in Appendix C, D, and E, respectively. We also report generalization performance and ablation studies of LRI in Appendix F. A summary of dataset statistics is shown in Table 3.

Backbone Models include DGCNN [52], Point Transformer [53], and EGNN [47] which have been widely used for scientific GDL [11, 56, 57].

Baseline interpretation methods include three masking-based methods BernMask, BernMask-P and PointMask, and two gradient-based methods GradGeo and GradGAM. Masking-based methods attempt to learn a mask $\in [0, 1]$ for each point and may help with testing existence importance of points. Among them, BernMask and BernMask-P are post-hoc methods extended from two previous methods on graph-structured data [58] and [59], respectively. BernMask and BernMask-P differ in the way they generate the masks, where BernMask-P utilizes a parameterized mask generator while BernMask optimizes a randomly initialized mask with no other parameterization. PointMask is an inherently interpretable model adopted from [60]. Thus, they are the baselines of LRI-Bernoulli. GradGeo is extended from [61] and checks the gradients w.r.t. geometric features, which may help with testing location importance, and thus is a baseline of LRI-Gaussian. GradGAM is extended from [62] and leverages the gradients w.r.t. the learned representations of points, which also reflects the importance of points to the prediction.

Metrics. On each dataset, the model is trained based on the prediction (binary classification) task. And all hyperparameters are tuned based on validation prediction performance for a fair comparison, because tuning on interpretation performance is impossible in a real-world setting. We compare interpretation labels ($\in \{0, 1\}$) of points with the learned point importance scores to measure interpretation performance. We report two metrics: interpretation ROC AUC and precision@m. Beyond interpretation, the model prediction performance is also measured in ROC AUC and reported in Sec. F.1, which is to make sure that all models are pre-trained well for those post-hoc baselines and to verify that LRI-induced models also have good prediction accuracy.

4.1 ActsTrack: End-to-end Pattern Recognition and Track Reconstruction

Here, we are to predict whether a collision event contains a $z \to \mu\mu$ decay. Each point in a point cloud sample is a detector hit where a particle passes through. We use the hits from the decay to test interpretation performance. We evaluate all methods on the data generated with a magnetic

Table 1: Interpretation performance on the four datasets. The **Bold**†, **Bold**, and Underline highlight the first, second, and third best results, respectively. All results are reported with mean $\pm$ std.

`ActsTrack`	DGCNN			Point Transformer			EGNN		
	ROC AUC	Prec@12	Prec@24	ROC AUC	Prec@12	Prec@24	ROC AUC	Prec@12	Prec@24
Random	50	21	21	50	21	21	50	21	21
GradGeo	65.92 ± 1.55	30.71 ± 1.46	30.20 ± 1.07	65.92 ± 1.61	31.76 ± 1.04	30.06 ± 1.32	67.57 ± 0.65	31.09 ± 1.37	30.87 ± 1.24
BernMask	68.85 ± 4.72	45.90 ± 7.14	42.72 ± 7.91	76.94 ± 1.99	71.17 ± 1.66	57.10 ± 3.22	50.94 ± 3.86	18.67 ± 2.06	20.00 ± 3.13
BernMask-P	88.16 ± 3.22	76.94 ± 11.30	66.71 ± 6.99	84.36 ± 2.64	73.24 ± 6.60	60.41 ± 5.44	30.81 ± 27.63	17.38 ± 32.08	13.92 ± 21.22
PointMask	49.85 ± 6.11	21.05 ± 4.67	21.54 ± 4.16	50.66 ± 0.91	22.43 ± 5.78	20.63 ± 3.63	49.55 ± 1.40	20.35 ± 1.14	20.06 ± 1.01
GradGAM	82.96 ± 2.42	84.29 ± 2.25	66.34 ± 2.56	83.07 ± 2.28	82.13 ± 0.92	64.64 ± 1.89	**79.44** ± 2.62	78.38 ± 1.96	52.55 ± 3.38
LRI-Bernoulli	**90.29** ± 1.39	**89.36** ± 1.20	**74.59** ± 1.28	**87.06** ± 2.49	**85.71** ± 0.99	**67.65** ± 1.22	78.57 ± 3.34	**81.56** ± 1.13	**55.29** ± 1.99
LRI-Gaussian	**95.49**† ± 0.34	**92.40**† ± 0.64	**79.87**† ± 0.72	**93.11**† ± 1.50	**88.39**† ± 4.36	**74.71**† ± 1.99	**94.34**† ± 0.70	**91.54**† ± 1.49	**76.78**† ± 0.81

`Tau3Mu`	DGCNN			Point Transformer			EGNN		
	ROC AUC	Prec@3	Prec@7	ROC AUC	Prec@3	Prec@7	ROC AUC	Prec@3	Prec@7
Random	50	35	35	50	35	35	50	35	35
GradGeo	**80.76** ± 0.31	68.86 ± 0.37	**56.05** ± 0.34	**79.62** ± 0.33	**67.96** ± 0.49	55.13 ± 0.24	78.05 ± 1.13	64.75 ± 2.09	54.28 ± 0.78
BernMask	54.28 ± 1.15	50.88 ± 1.64	42.21 ± 0.94	30.00 ± 0.40	12.48 ± 0.58	18.81 ± 0.27	72.27 ± 2.66	61.27 ± 3.68	51.91 ± 1.63
BernMask-P	54.99 ± 14.06	43.56 ± 13.59	39.23 ± 9.49	76.46 ± 4.05	59.39 ± 4.78	53.41 ± 3.30	70.99 ± 5.59	56.36 ± 6.04	50.06 ± 4.44
PointMask	52.92 ± 2.56	39.55 ± 2.27	36.71 ± 1.48	57.33 ± 2.28	44.00 ± 1.91	38.53 ± 2.41	55.90 ± 5.39	39.92 ± 8.20	37.05 ± 4.74
GradGAM	68.42 ± 4.04	51.82 ± 6.02	45.78 ± 3.60	**80.91**† ± 0.35	61.14 ± 1.46	54.36 ± 0.67	75.84 ± 1.45	62.03 ± 2.41	53.19 ± 1.32
LRI-Bernoulli	77.88 ± 1.03	**70.66** ± 0.90	55.94 ± 0.77	77.72 ± 1.52	67.73 ± 2.59	**55.74** ± 1.15	**78.71** ± 0.66	**65.99** ± 0.84	**55.98** ± 0.57
LRI-Gaussian	**81.38**† ± 0.62	**73.13**† ± 1.10	**58.28**† ± 0.59	79.58 ± 0.66	**70.32**† ± 0.76	**57.05**† ± 0.53	**80.02**† ± 0.39	**71.20**† ± 0.93	**57.07** ± 0.41

`SynMol`	DGCNN			Point Transformer			EGNN		
	ROC AUC	Prec@5	Prec@8	ROC AUC	Prec@5	Prec@8	ROC AUC	Prec@5	Prec@8
Random	50	31	31	50	31	31	50	31	31
GradGeo	72.10 ± 9.66	59.59 ± 11.05	50.30 ± 8.70	76.94 ± 1.43	62.30 ± 0.78	55.29 ± 0.87	73.49 ± 5.23	61.85 ± 5.26	50.46 ± 3.95
BernMask	49.69 ± 9.22	34.37 ± 9.32	32.15 ± 7.64	25.28 ± 3.52	6.85 ± 1.57	8.65 ± 1.14	59.76 ± 9.09	49.96 ± 7.56	40.72 ± 7.25
BernMask-P	70.51 ± 39.52	63.02 ± 36.13	52.93 ± 30.14	87.23 ± 6.07	75.39 ± 9.74	63.00 ± 7.31	90.00 ± 7.85	**85.52** ± 5.75	**68.94** ± 6.37
PointMask	74.22 ± 3.31	71.54 ± 4.27	55.18 ± 2.86	72.03 ± 2.10	60.13 ± 2.57	51.11 ± 1.43	65.43 ± 6.63	53.89 ± 2.25	48.11 ± 3.05
GradGAM	81.98 ± 5.54	78.80 ± 6.67	59.86 ± 5.86	85.54 ± 1.19	80.24 ± 1.98	64.38 ± 1.49	57.00 ± 5.52	48.07 ± 8.56	41.30 ± 5.08
LRI-Bernoulli	**96.03** ± 1.54	**87.11** ± 4.51	**74.57** ± 1.57	**91.69** ± 1.52	**82.72** ± 2.20	**68.37** ± 1.11	**90.64** ± 3.30	71.96 ± 5.97	68.08 ± 4.18
LRI-Gaussian	**99.02**† ± 0.36	**97.72**† ± 0.94	**77.04**† ± 0.43	**95.35**† ± 1.02	**87.09**† ± 1.97	**72.26**† ± 1.40	**97.28**† ± 0.65	**91.52**† ± 1.28	**74.05**† ± 1.18

`PLBind`	DGCNN			Point Transformer			EGNN		
	ROC AUC	Prec@20	Prec@40	ROC AUC	Prec@20	Prec@40	ROC AUC	Prec@20	Prec@40
Random	50	45	45	50	45	45	50	45	45
GradGeo	52.83 ± 4.63	55.68 ± 2.47	53.79 ± 1.83	58.68 ± 2.83	59.30 ± 3.13	57.85 ± 3.57	**57.78**† ± 2.61	61.00 ± 2.24	60.11 ± 1.76
BernMask	48.18 ± 4.14	48.36 ± 3.32	48.00 ± 3.40	**59.73**† ± 2.33	59.30 ± 3.09	58.73 ± 2.90	49.83 ± 2.17	40.34 ± 3.96	41.99 ± 2.32
BernMask-P	48.88 ± 5.66	42.70 ± 8.37	42.46 ± 7.88	56.47 ± 2.77	56.86 ± 3.79	54.53 ± 4.64	51.96 ± 6.80	60.68 ± 7.95	57.69 ± 6.41
PointMask	51.38 ± 3.12	45.36 ± 1.91	45.22 ± 1.52	52.92 ± 3.83	44.34 ± 3.50	44.50 ± 4.52	50.00 ± 0.00	45.10 ± 0.00	45.00 ± 0.00
GradGAM	**53.76** ± 3.38	55.50 ± 4.83	54.23 ± 4.42	56.51 ± 3.32	56.54 ± 6.59	54.46 ± 5.65	49.73 ± 2.18	56.92 ± 6.63	53.96 ± 3.53
LRI-Bernoulli	**55.47**† ± 2.06	**67.56**† ± 5.38	**61.02**† ± 5.68	**59.53** ± 1.94	**72.98**† ± 3.85	**67.33**† ± 2.08	**57.07** ± 3.09	**73.22**† ± 2.32	**66.89** ± 2.07
LRI-Gaussian	51.81 ± 3.24	**63.88** ± 3.18	**60.37** ± 3.09	54.05 ± 2.94	**62.76** ± 5.93	**60.44** ± 5.62	50.32 ± 3.80	**72.64** ± 2.04	**69.40**† ± 1.44

Table 2: The angle (°, mean±std) between the velocity and the first principal component of $\mathbf{\Sigma}_v$ v.s. between the velocity and the gradient of $\mathbf{r}_v$ in x-y space under different magnetic field strengths (T).

	2T	4T	6T	8T	10T	12T	14T	16T	18T	20T
Random	45	45	45	45	45	45	45	45	45	45
GradGeo	67.11 ± 2.45	71.15 ± 1.29	55.21 ± 2.12	59.43 ± 2.82	59.73 ± 2.54	52.78 ± 3.18	51.99 ± 3.24	50.77 ± 1.60	51.99 ± 3.89	53.74 ± 5.34
LRI-Gaussian	5.09 ± 0.97	5.17 ± 0.42	5.65 ± 1.05	6.67 ± 1.17	5.99 ± 1.71	6.66 ± 1.25	7.50 ± 2.37	7.19 ± 1.00	7.57 ± 1.20	7.89 ± 1.03

field $B = 2$T parallel to the z axis (see Fig. 1a). Table 1 shows the results, where LRI-Gaussian works consistently the best on all backbones and metrics, and LRI-Bernoulli achieves the second best performance. This indicates that both the *existence* and *locations* of those detector hits left by the μ's are important to predictions. GradGAM is also a competitive baseline, which even outperforms both BernMask and BernMask-P. While BernMask-P performs decently on two backbones, it fails to provide interpretability for EGNN and its results have high variances, probably due to the unstable issue of post-hoc interpretations, as described in [24]. BernMask, PointMask, and GradGeo seem unable to provide valid interpretability for `ActsTrack`.

Fine-grained Geometric Patterns. We find LRI-Gaussian can discover fine-grained geometric patterns. Intuitively, slight perturbation of each point along the underlying track direction will affects the model prediction less than the same level of perturbation orthogonal to the track. Therefore, the principal component of $\mathbf{\Sigma}_v$ in x-y space, i.e., the space orthogonal to the direction of the background magnetic field, can give an estimation of the track direction at v. Table 2 provides the evaluation of track direction (velocity direction) estimation based on analyzing $\mathbf{\Sigma}_v$. Here, we test the background magnetic field changing from 2T to 20T. LRI-Gaussian is far more accurate than GradGeo. The latter

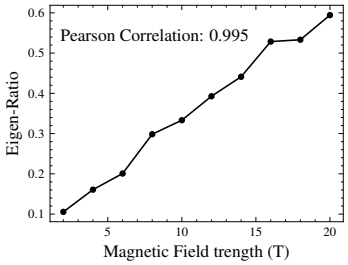


Figure 3: Eigen-ratio of $\mathbf{\Sigma}_v$ v.s. magnetic field strength B (T).

uses the gradients of different coordinates to compute the most sensitive direction. Moreover, the ratio between the lengths of two principal components of $\mathbf{\Sigma}_v$ in x-y space gives an estimation of the curvature of the track at v, which is proportion to the strength of the magnetic field B up to a constant multiplier (due to the Lorentz force $F \propto B$). Therefore, we can estimate B by analyzing $\mathbf{\Sigma}_v$. Fig. 3 shows this approach provides an almost accurate estimation of B up to a constant multiplier.

4.2 Tau3Mu: $\tau \to \mu\mu\mu$ Decay Detection in Proton-Proton Collisions

This task is to predict whether a proton-proton collision event contains a $\tau \to \mu\mu\mu$ decay, which is similar to `ActsTrack`, while in `Tau3Mu` the μ's are a lot softer, and only the hits from the first layer of detectors are used. Each point in a point cloud sample is a detector hit associated with a local bending angle and a 2D coordinate in the pseudorapidity-azimuth (η-ϕ) space. We use the hits generated from this decay to test interpretation performance. As shown in Table 1, LRI-Gaussian still works the best. LRI-Bernoulli and GradGeo are close, and both are the second best. While GradGAM still works well on some backbones, all masking-based methods do not perform well.

4.3 SynMol: Molecular Property Prediction with Synthetic Properties

This task is to predict whether a molecule contains both functional groups *branched alkanes* and *carbonyl*, which together give certain synthetic properties [33, 34]. Each point in a point cloud sample is an atom associated with a 3D coordinate and a categorical feature indicating the atom type. We use the atoms in these two functional groups to test interpretation performance. As shown in Table 1, LRI-Gaussian performs consistently the best by only perturbing geometric features in molecules, and LRI-Bernoulli works the second best and achieves comparable performance with LRI-Gaussian on Point Transformer. This shows that both the existence and locations of atoms are critical and further validates the benefit of using 3D representations of molecules in the tasks like molecular property prediction. Among other methods, GradGAM, BernMask-P and PointMask are unstable and can only provide some interpretability for one or two backbones, while GradGeo and BernMask seem to fail to perform well on `SynMol`.

4.4 PLBind: Protein-ligand Binding Affinity Prediction

This task is to predict whether a protein-ligand pair is of affinity $K_D < 10$ nM. Each point in a protein is a amino acid associated with a 3D coordinate, a categorical feature indicating the amino acid type, and two scalar features. Each point in a ligand is an atom associated with a 3D coordinate, a categorical feature indicating the atom type, and a scalar feature. Different from other datasets, each sample in `PLBind` contains two sets of points. So, for each sample, two encoders will be used to encode the ligand and the protein separately, and the obtained two embeddings will be added to make a prediction. As shown in Table 1, LRI-Bernoulli outperforms all other methods, while LRI-Gaussian achieves comparable performance on EGNN. This might indicate that to make good predictions on `PLBind`, the existence of certain groups of amino acids is more important than their exact locations. Interestingly, all other methods do not seem to perform well on `PLBind`. Moreover, all methods have low ROC AUC, which suggests only a part of but not the entire binding site is important to decide the binding affinity.

5 Conclusion

This work systematically studies interpretable GDL models by proposing a framework *Learnable Randomness Injection* (LRI) and four datasets with ground-truth interpretation labels from real-world scientific applications. We have studied interpretability in GDL from the perspectives of *existence importance* and *location importance* of points, and instantiated LRI with LRI-Bernoulli and LRI-Gaussian to test the two types of importance, respectively. We observe LRI-induced models provide interpretation best aligning with scientific facts, especially LRI-Gaussian that tests location importance. Grounded on the IB principle, LRI never degrades model prediction performance, and may often improve it when there exist distribution shifts between the training and test scenarios.

References

[1] D. Guest, K. Cranmer, and D. Whiteson, "Deep learning and its application to lhc physics," *Annual Review of Nuclear and Particle Science*, vol. 68, no. 1, 2018.

[2] R. Wang, X. Fang, Y. Lu, and S. Wang, "The pdbbind database: Collection of binding affinities for protein- ligand complexes with known three-dimensional structures," *Journal of medicinal chemistry*, vol. 47, no. 12, pp. 2977–2980, 2004.

[3] R. Wang, X. Fang, Y. Lu, C.-Y. Yang, and S. Wang, "The pdbbind database: methodologies and updates," *Journal of medicinal chemistry*, vol. 48, no. 12, pp. 4111–4119, 2005.

[4] G. E. Tusnady and I. Simon, "Principles governing amino acid composition of integral membrane proteins: application to topology prediction," *Journal of molecular biology*, vol. 283, no. 2, pp. 489–506, 1998.

[5] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. A. Khalek, A. A. Abdelalim, R. Aben, B. Abi, M. Abolins, O. AbouZeid *et al.*, "Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc," *Physics Letters B*, vol. 716, no. 1, pp. 1–29, 2012.

[6] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, "Machine learning for molecular and materials science," *Nature*, vol. 559, no. 7715, pp. 547–555, 2018.

[7] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," *Reviews of Modern Physics*, vol. 91, no. 4, p. 045002, 2019.

[8] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[9] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv preprint arXiv:2104.13478*, 2021.

[10] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 021001, 2020.

[11] H. Qu and L. Gouskos, "Jet tagging via particle clouds," *Physical Review D*, vol. 101, no. 5, p. 056019, 2020.

[12] P. Gainza, F. Sverrisson, F. Monti, E. Rodola, D. Boscaini, M. Bronstein, and B. Correia, "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning," *Nature Methods*, vol. 17, no. 2, pp. 184–192, 2020.

[13] R. J. Townshend, S. Eismann, A. M. Watkins, R. Rangan, M. Karelina, R. Das, and R. O. Dror, "Geometric deep learning of rna structure," *Science*, vol. 373, no. 6558, pp. 1047–1051, 2021.

[14] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *International conference on machine learning*. PMLR, 2016, pp. 2990–2999.

[15] A. Bogatskiy, B. Anderson, J. Offermann, M. Roussi, D. Miller, and R. Kondor, "Lorentz group equivariant neural network for particle physics," in *International Conference on Machine Learning*. PMLR, 2020, pp. 992–1002.

[16] R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke, "Explainable machine learning for scientific insights and discoveries," *Ieee Access*, vol. 8, pp. 42 200–42 216, 2020.

[17] K. Y. Gao, A. Fokoue, H. Luo, A. Iyengar, S. Dey, P. Zhang *et al.*, "Interpretable drug target prediction using deep neural representation." in *IJCAI*, vol. 2018, 2018, pp. 3371–3377.

[18] M. Karimi, D. Wu, Z. Wang, and Y. Shen, "Deepaffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks," *Bioinformatics*, vol. 35, no. 18, pp. 3329–3338, 2019.

[19] ——, "Explainable deep relational networks for predicting compound–protein affinities and contacts," *Journal of chemical information and modeling*, vol. 61, no. 1, pp. 46–66, 2020.

[20] B. Nachman and C. Shimmin, "Ai safety for high energy physics," *arXiv preprint arXiv:1910.08606*, 2019.

[21] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[22] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "The dangers of post-hoc interpretability: Unjustified counterfactual explanations," *arXiv preprint arXiv:1907.09294*, 2019.

[23] S. Bordt, M. Finck, E. Raidl, and U. von Luxburg, "Post-hoc explanations fail to achieve their purpose in adversarial contexts," *arXiv preprint arXiv:2201.10295*, 2022.

[24] S. Miao, M. Liu, and P. Li, "Interpretable and generalizable graph learning via stochastic attention mechanism," in *International Conference on Machine Learning*. PMLR, 2022, pp. 15 524–15 543.

[25] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[26] R. Schulte, V. Bashkirov, T. Li, Z. Liang, K. Mueller, J. Heimann, L. R. Johnson, B. Keeney, H.-W. Sadrozinski, A. Seiden *et al.*, "Conceptual design of a proton computed tomography system for applications in proton radiation therapy," *IEEE Transactions on Nuclear Science*, vol. 51, no. 3, pp. 866–872, 2004.

[27] M. Thomson, *Modern particle physics*. Cambridge University Press, 2013.

[28] X. Ai, C. Allaire, N. Calace, A. Czirkos, M. Elsing, I. Ene, R. Farkas, L.-G. Gagnon, R. Garg, P. Gessinger *et al.*, "A common tracking software project," *Computing and Software for Big Science*, vol. 6, no. 1, pp. 1–23, 2022.

[29] R. Oerter, *The theory of almost everything: The standard model, the unsung triumph of modern physics*. Penguin, 2006.

[30] P. Blackstone, M. Fael, and E. Passemar, "$\tau \to \mu\mu\mu$ at a rate of one out of 10^{14} tau decays?" *The European Physical Journal C*, vol. 80, no. 6, jun 2020.

[31] L. Calibbi and G. Signorelli, "Charged lepton flavour violation: an experimental and theoretical introduction," *La Rivista del Nuovo Cimento*, vol. 41, no. 2, pp. 71–174, 2018.

[32] A. Collaboration *et al.*, "Search for charged-lepton-flavour violation in z-boson decays with the atlas detector," *arXiv preprint arXiv:2010.02566*, 2020.

[33] K. McCloskey, A. Taly, F. Monti, M. P. Brenner, and L. J. Colwell, "Using attribution to decode binding mechanism in neural network models for chemistry," *Proceedings of the National Academy of Sciences*, vol. 116, no. 24, pp. 11 624–11 629, 2019.

[34] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Wang, W. Qian, K. McCloskey, L. Colwell, and A. Wiltschko, "Evaluating attribution for graph neural networks," *Advances in neural information processing systems*, vol. 33, pp. 5898–5910, 2020.

[35] C. Wang and Y. Zhang, "Improving scoring-docking-screening powers of protein–ligand scoring functions using random forest," *Journal of computational chemistry*, vol. 38, no. 3, pp. 169–177, 2017.

[36] M. Held, P. Metzner, J.-H. Prinz, and F. Noé, "Mechanisms of protein-ligand association and its modulation by protein mutations," *Biophysical journal*, vol. 100, no. 3, pp. 701–710, 2011.

[37] X. Du, Y. Li, Y.-L. Xia, S.-M. Ai, J. Liang, P. Sang, X.-L. Ji, and S.-Q. Liu, "Insights into protein–ligand interactions: mechanisms, models, and methods," *International journal of molecular sciences*, vol. 17, no. 2, p. 144, 2016.

[38] Z. Cang and G.-W. Wei, "Integration of element specific persistent homology and machine learning for protein-ligand binding affinity prediction," *International journal for numerical methods in biomedical engineering*, vol. 34, no. 2, p. e2914, 2018.

[39] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

[40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[41] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[42] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[43] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[44] F. Fuchs, D. Worrall, V. Fischer, and M. Welling, "Se (3)-transformers: 3d roto-translation equivariant attention networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1970–1981, 2020.

[45] J. Klicpera, J. Groß, and S. Günnemann, "Directional message passing for molecular graphs," *arXiv preprint arXiv:2003.03123*, 2020.

[46] D. Beaini, S. Passaro, V. Létourneau, W. Hamilton, G. Corso, and P. Liò, "Directional graph networks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 748–758.

[47] V. G. Satorras, E. Hoogeboom, and M. Welling, "E (n) equivariant graph neural networks," in *International conference on machine learning*. PMLR, 2021, pp. 9323–9332.

[48] K. Schütt, O. Unke, and M. Gastegger, "Equivariant message passing for the prediction of tensorial properties and molecular spectra," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9377–9388.

[49] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," *Advances in neural information processing systems*, vol. 30, 2017.

[50] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, "Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds," *arXiv preprint arXiv:1802.08219*, 2018.

[51] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630.

[52] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[53] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268.

[54] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[55] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[56] K. Atz, F. Grisoni, and G. Schneider, "Geometric deep learning on molecular representations," *Nature Machine Intelligence*, vol. 3, no. 12, pp. 1023–1032, 2021.

[57] L. Gagliardi, A. Raffo, U. Fugacci, S. Biasotti, W. Rocchia, H. Huang, B. B. Amor, Y. Fang, Y. Zhang, X. Wang *et al.*, "Shrec 2022: Protein–ligand binding site recognition," *Computers & Graphics*, vol. 107, pp. 20–31, 2022.

[58] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[59] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," *Advances in neural information processing systems*, vol. 33, pp. 19 620–19 631, 2020.

[60] S. A. Taghanaki, K. Hassani, P. K. Jayaraman, A. H. Khasahmadi, and T. Custis, "Point-mask: Towards interpretable and bias-resilient point cloud processing," *arXiv preprint arXiv:2007.04525*, 2020.

[61] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International conference on machine learning*. PMLR, 2017, pp. 3145–3153.

[62] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[63] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[64] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 3319–3328.

[65] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," in *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2018, pp. 839–847.

[66] M. S. Schlichtkrull, N. De Cao, and I. Titov, "Interpreting graph neural networks for nlp with differentiable edge masking," *arXiv preprint arXiv:2010.00577*, 2020.

[67] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 241–12 252.

[68] J. Chen, L. Song, M. Wainwright, and M. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *International Conference on Machine Learning*. PMLR, 2018, pp. 883–892.

[69] J. Yoon, J. Jordon, and M. van der Schaar, "Invase: Instance-wise variable selection using neural networks," in *International Conference on Learning Representations*, 2018.

[70] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[71] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[72] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[73] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan, "L-shapley and c-shapley: Efficient model interpretation for structured data," *arXiv preprint arXiv:1808.02610*, 2018.

[74] M. Ancona, C. Oztireli, and M. Gross, "Explaining deep neural networks with a polynomial time algorithm for shapley value approximation," in *International Conference on Machine Learning*. PMLR, 2019, pp. 272–281.

[75] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable ai for trees," *Nature machine intelligence*, vol. 2, no. 1, pp. 56–67, 2020.

[76] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[77] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[78] S. Serrano and N. A. Smith, "Is attention interpretable?" *arXiv preprint arXiv:1906.03731*, 2019.

[79] S. Jain and B. C. Wallace, "Attention is not explanation," *arXiv preprint arXiv:1902.10186*, 2019.

[80] B. Bai, J. Liang, G. Zhang, H. Li, K. Bai, and F. Wang, "Why attentions may not be interpretable?" in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 25–34.

[81] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[82] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, "This looks like that: deep learning for interpretable image recognition," *Advances in neural information processing systems*, vol. 32, 2019.

[83] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He, "Graph information bottleneck for subgraph recognition," *arXiv preprint arXiv:2010.05563*, 2020.

[84] Q. Sun, J. Li, H. Peng, J. Wu, X. Fu, C. Ji, and S. Y. Philip, "Graph structure learning with variational information bottleneck," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 4165–4174.

[85] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.

[86] S. Chang, Y. Zhang, M. Yu, and T. Jaakkola, "Invariant rationalization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1448–1458.

[87] D. Krueger, E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, D. Zhang, R. Le Priol, and A. Courville, "Out-of-distribution generalization via risk extrapolation (rex)," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5815–5826.

[88] Y.-X. Wu, X. Wang, A. Zhang, X. He, and T.-S. Chua, "Discovering invariant rationales for graph neural networks," *arXiv preprint arXiv:2201.12872*, 2022.

[89] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *International Conference on Learning Representations*, 2017.

[90] A. Achille and S. Soatto, "Emergence of invariance and disentanglement in deep representations," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 1947–1980, 2018.

[91] T. Wu, H. Ren, P. Li, and J. Leskovec, "Graph information bottleneck," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 437–20 448, 2020.

[92] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, "An introduction to pythia 8.2," *Computer physics communications*, vol. 191, pp. 159–177, 2015.

[93] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman, "Zinc: a free tool to discover chemistry for biology," *Journal of chemical information and modeling*, vol. 52, no. 7, pp. 1757–1768, 2012.

[94] S. Riniker and G. A. Landrum, "Better informed distance geometry: using what we know to improve conformation generation," *Journal of chemical information and modeling*, vol. 55, no. 12, pp. 2562–2574, 2015.

[95] T. A. Halgren, "Mmff vi. mmff94s option for energy minimization studies," *Journal of computational chemistry*, vol. 20, no. 7, pp. 720–729, 1999.

[96] Z. Liu, M. Su, L. Han, J. Liu, Q. Yang, Y. Li, and R. Wang, "Forging the basis for developing protein–ligand interaction scoring functions," *Accounts of chemical research*, vol. 50, no. 2, pp. 302–309, 2017.

[97] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.

[98] M. Liu, Y. Luo, K. Uchino, K. Maruhashi, and S. Ji, "Generating 3d molecules for target protein binding," *arXiv preprint arXiv:2204.09410*, 2022.

[99] R. A. Laskowski, J. Jabłońska, L. Pravda, R. S. Vařeková, and J. M. Thornton, "Pdbsum: Structural summaries of pdb entries," *Protein science*, vol. 27, no. 1, pp. 129–134, 2018.

[100] H. Stärk, O. Ganea, L. Pattanaik, R. Barzilay, and T. Jaakkola, "Equibind: Geometric deep learning for drug binding structure prediction," in *International Conference on Machine Learning*. PMLR, 2022, pp. 20 503–20 521.

[101] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[102] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[103] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[104] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[105] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[106] J. Gildenblat and contributors, "Pytorch library for cam methods," https://github.com/jacobgil/pytorch-grad-cam, 2021.

A Related Work

We review two categories of methods that can provide interpretability in the following.

Post-hoc Interpretation Methods. Interpretation methods falling into this category assume a pre-trained model is given and attempts to further analyze it to provide post-hoc interpretation. Among them, gradient-based methods [63, 62, 64, 61, 65] may be extended to interpret geometric data by checking the gradients w.r.t. the input features or intermediate embeddings of each point. Some methods to interpret graph neural networks can be applied to geometric data [58, 59, 66, 67]. However, these methods need to mask graph structures pre-constructed by geometric features and cannot fully evaluate the effectiveness of geometric features. Among other methods, [68, 69] study pattern selection for regular data, [70–72] utilize a local surrogate model, and [71, 73–75] leverage the shapley value to evaluate feature importance. These methods either cannot utilize geometric features or cannot be easily applied to irregular geometric data.

Inherently Interpretable Models. Although vanilla attention mechanisms [76, 77] were widely used for inherent interpretability, multiple recent studies show that they cannot provide reliable interpretation, especially for data with irregular structures [78, 79, 58, 59]. So, some works focusing on improving the attention mechanism for better interpretability [80, 24], some propose to identify representative prototypes during training [81, 82], and some methods [60, 83, 84] adopt the information bottleneck principle [25]. However, all these methods cannot analyze geometric features in GDL. Along another line, although invariant learning methods [85–88] based on causality analysis may also provide some interpretability, these methods are typically of great complexity and seem to provide subpar interpretability even on graph-structured data without geometric features [24].

B Connecting LRI and the Information Bottleneck Principle.

Our objectives Eq. 1 and Eq. 2 are essentially variational upper bounds of the information bottleneck (IB) principle [25, 89] whose goal is to reduce the mutual information between $\mathcal{C}$ and $\tilde{\mathcal{C}}$ while keeping the mutual information between $\tilde{\mathcal{C}}$ and the label, i.e., $\min -I(\tilde{\mathcal{C}}; y) + \beta I(\tilde{\mathcal{C}}; \mathcal{C})$. We provide derivations in the following. Grounded on the IB principle, LRI tends to extract minimal sufficient information to make predictions and can be more robust to distribution shifts between training and test datasets [90, 91, 24].

B.1 Variational Bounds of the IB Principle

Let's assume the sample $\mathcal{C} \sim \mathbb{P}_{\mathbb{C}}$. We ignore the $\tilde{\mathcal{G}}$ in our objectives to keep the notation simple, then, the IB objective is:

$$\min -I(\tilde{\mathcal{C}}; y) + \beta I(\tilde{\mathcal{C}}; \mathcal{C}), \tag{3}$$

where y is the label, $\tilde{\mathcal{C}}$ is the perturbed sample, and $I(.;.)$ denotes the mutual information between two random variables.

For the first term $-I(\tilde{\mathcal{C}}; y)$, by definition we have:

$$-I(\tilde{\mathcal{C}}; y) = -\mathbb{E}_{\tilde{\mathcal{C}},y}\left[\log \frac{\mathbb{P}(y \mid \tilde{\mathcal{C}})}{\mathbb{P}(y)}\right]. \tag{4}$$

We introduce a variational approximation $\mathbb{P}_\theta(y \mid \tilde{\mathcal{C}})$ for $\mathbb{P}(y \mid \tilde{\mathcal{C}})$ as it is intractable. Then, we yield a variational upper bound:

$$\begin{aligned}
-I(\tilde{\mathcal{C}}; y) &= -\mathbb{E}_{\tilde{\mathcal{C}},y}\left[\log \frac{\mathbb{P}_\theta(y \mid \tilde{\mathcal{C}})}{\mathbb{P}(y)}\right] - \mathbb{E}_{\tilde{\mathcal{C}}}\left[D_{\mathrm{KL}}(\mathbb{P}(y \mid \tilde{\mathcal{C}}) \| \mathbb{P}_\theta(y \mid \tilde{\mathcal{C}}))\right] \\
&\leq -\mathbb{E}_{\tilde{\mathcal{C}},y}\left[\log \frac{\mathbb{P}_\theta(y \mid \tilde{\mathcal{C}})}{\mathbb{P}(y)}\right] \\
&= -\mathbb{E}_{\tilde{\mathcal{C}},y}\left[\log \mathbb{P}_\theta(y \mid \tilde{\mathcal{C}})\right] - H(y),
\end{aligned} \tag{5}$$

where $H(y)$ is a constant, therefore, minimizing the cross-entropy loss $\mathcal{L}_{CE}(f_\theta(\tilde{\mathcal{C}}), y)$ is equivalent to minimizing a variational upper bound of $-I(\tilde{\mathcal{C}}; y)$.

For the second term $I(\tilde{\mathcal{C}}; \mathcal{C})$, because $\tilde{\mathcal{C}} = g_\phi(\mathcal{C})$, i.e., $\tilde{\mathcal{C}}$ is given by a model parameterized by ϕ, by definition, we have:

$$I(\tilde{\mathcal{C}}; \mathcal{C}) = \mathbb{E}_{\tilde{\mathcal{C}},\mathcal{C}}\left[\log \frac{\mathbb{P}_\phi(\tilde{\mathcal{C}} \mid \mathcal{C})}{\mathbb{P}(\tilde{\mathcal{C}})}\right]. \tag{6}$$

As $\mathbb{P}(\tilde{\mathcal{C}})$ is intractable, we introduce a variational approximation $\mathbb{Q}(\tilde{\mathcal{C}})$. Then, we yield a variational upper bound:

$$\begin{aligned}
I(\tilde{\mathcal{C}}; \mathcal{C}) &= \mathbb{E}_{\tilde{\mathcal{C}},\mathcal{C}}\left[\log \frac{\mathbb{P}_\phi(\tilde{\mathcal{C}} \mid \mathcal{C})}{\mathbb{Q}(\tilde{\mathcal{C}})}\right] - D_{\mathrm{KL}}\left(\mathbb{P}(\tilde{\mathcal{C}}) \| \mathbb{Q}(\tilde{\mathcal{C}})\right) \\
&\leq \mathbb{E}_{\mathcal{C}}\left[D_{\mathrm{KL}}\left(\mathbb{P}_\phi(\tilde{\mathcal{C}} \mid \mathcal{C}) \| \mathbb{Q}(\tilde{\mathcal{C}})\right)\right].
\end{aligned} \tag{7}$$

For LRI-Bernoulli, g_ϕ takes as input $\mathcal{C} = (\mathcal{V}, \mathbf{X}, \mathbf{r})$ and first outputs $p_v \in [0, 1]$ for each point $v \in \mathcal{V}$. Then, it samples $m_v \sim \text{Bern}(p_v)$ and yields $\tilde{\mathcal{C}}$ by removing all points with $m_v = 0$ in $\mathcal{C}$. This procedure gives $\mathbb{P}_\phi(\tilde{\mathcal{C}} \mid \mathcal{C}) = \prod_{v \in \mathcal{V}} \mathbb{P}(m_v | p_v)$. In this case, we define $\mathbb{Q}(\tilde{\mathcal{C}})$ as follows. For every point cloud $\mathcal{C} \sim \mathbb{P}_{\mathbb{C}}$, we sample $m'_v \sim \text{Bern}(\alpha)$, where $\alpha \in [0, 1]$ is a hyperparameter. We remove all points in $\mathcal{C}$ and add points when their $m'_v = 1$. This procedure gives $\mathbb{Q}(\tilde{\mathcal{C}}) = \sum_{\mathcal{C}} \mathbb{P}(\mathbf{m}' \mid \mathcal{C}) \mathbb{P}_{\mathbb{C}}(C)$. As $\mathbf{m}'$ is independent from $\mathcal{C}$ given its size n, $\mathbb{Q}(\tilde{\mathcal{C}}) = \sum_n \mathbb{P}(\mathbf{m}' | n) \mathbb{P}_{\mathbb{C}}(\mathcal{C} = n) = \mathbb{P}(n) \prod_{v=1}^{n} \mathbb{P}(m'_v)$, where $\mathbb{P}(n)$ is a constant. Then, we yield:

$$D_{\mathrm{KL}}\left(\mathbb{P}_\phi(\tilde{\mathcal{C}} \mid \mathcal{C}) \| \mathbb{Q}(\tilde{\mathcal{C}})\right) = \sum_{v \in \mathcal{V}} D_{\mathrm{KL}}(\text{Bern}(p_v) \| \text{Bern}(\alpha)) + c(n, \alpha), \tag{8}$$

where $c(n, \alpha)$ is a constant, therefore, minimizing the second term of LRI-Bernoulli is equivalent to minimizing a variational upper bound of $I(\tilde{\mathcal{C}}; \mathcal{C})$. Now, we can conclude that the objective of LRI-Bernoulli is a variational upper bound of the IB principle.

For LRI-Gaussian, g_ϕ now takes as input $\mathcal{C} = (\mathcal{V}, \mathbf{X}, \mathbf{r})$ and first outputs a covariance matrix $\mathbf{\Sigma}_v \in \mathbb{R}^{3 \times 3}$ for each point $v \in \mathcal{V}$. Then, it samples $\epsilon_v \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_v)$ and yields $\tilde{\mathcal{C}} = (\mathcal{V}, \mathbf{X}, \mathbf{r} + \epsilon)$, where ϵ is a vector containing ϵ_v for each point. This procedure gives $\mathbb{P}_\phi(\tilde{\mathcal{C}} \mid \mathcal{C}) = \prod_{v \in \mathcal{V}} \mathbb{P}(\epsilon_v | \mathbf{\Sigma}_v)$. In this case, we define $\mathbb{Q}(\tilde{\mathcal{C}})$ as follows. For every point cloud $\mathcal{C} \sim \mathbb{P}_{\mathbb{C}}$, we sample $\epsilon'_v \sim \mathcal{N}(\mathbf{0}, \sigma \mathbf{I})$ and yield $\tilde{\mathcal{C}} = (\mathcal{V}, \mathbf{X}, \mathbf{r} + \epsilon')$, where σ is a hyperparameter. Similarly, we obtain $\mathbb{Q}(\tilde{\mathcal{C}}) = \mathbb{P}(n) \prod_{v=1}^{n} \mathbb{P}(\epsilon'_v)$, where $\mathbb{P}(n)$ is a constant. Finally, for this case, we have:

$$D_{\mathrm{KL}}\left(\mathbb{P}_\phi(\tilde{\mathcal{C}} \mid \mathcal{C}) \| \mathbb{Q}(\tilde{\mathcal{C}})\right) = \sum_{v \in \mathcal{V}} D_{\mathrm{KL}}(\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_v) \| \mathcal{N}(\mathbf{0}, \sigma \mathbf{I}) + c(n, \alpha), \tag{9}$$

where $c(n, \alpha)$ is a constant, therefore, minimizing the second term of LRI-Gaussian is equivalent to minimizing a variational upper bound of $I(\tilde{\mathcal{C}}; \mathcal{C})$. Hence, the objective of LRI-Gaussian is also a variational upper bound of the IB principle.

Table 3: Statistics of the four datasets.

	# Classes	# Features in $\mathbf{X}$	# Dimensions in $\mathbf{r}$	# Samples	Avg. # Points/Sample	Avg. # Important Points/Sample	Class Ratio	Split Scheme	Split Ratio
`ActsTrack`	2	0	3	3241	109.1	22.8	39/61	Random	70/15/15
`Tau3Mu`	2	1	2	129687	16.9	5.5	24/76	Random	70/15/15
`SynMol`	2	1	3	8663	21.9	6.6	18/82	Patterns	78/11/11
`PLBind`	2	3	3	10891	339.8	132.2	29/71	Time	92/6/2

C Dataset Collection

We describe how we collect the four datasets in this section. Table 3 shows the statistics of the four datasets.

Basic Settings. 1) All coordinates in the four datasets are centered at the origin. 2) Only positive samples in our datasets have ground-truth interpretation labels, so we only evaluate interpretation performance on positive samples. 3) For any pair of points v and u in the four datasets, they have an edge feature of $(\|\mathbf{r}_v - \mathbf{r}_u\|, \frac{\mathbf{r}_v - \mathbf{r}_u}{\|\mathbf{r}_v - \mathbf{r}_u\|})$ if they are connected in the constructed k-nn graph.

ActsTrack. $z \to \mu\mu$ events are simulated with Pythia8 generator [92] overlaid with soft QCD pileup events, and particle tracks are simulated using Acts Common Tracking Software [28]. For all samples, ten pileup interactions are generated with a center-of-mass energy of 14TeV, and the additional hard scatter interaction is only generated for positive samples. Particle tracks are simulated with a magnetic field parallel to the z axis, and the default `ActsTrack` is simulated with $B = 2$T . To make sure both μ's from the decay are properly measured, we calculate the invariant mass m_{ij} for every pair of μ's in the generated data using Eq. 10 so that every positive sample in our dataset has at least a pair of μ' with an invariant mass close to 91.19 GeV, i.e., the mass of the z bosons.

$$\begin{aligned}\frac{1}{2}m_{ij}^2 = m^2 + \Big(\sqrt{m^2 + p_{x,i}^2 + p_{y,i}^2 + p_{z,i}^2} \cdot \sqrt{m^2 + p_{x,j}^2 + p_{y,j}^2 + p_{z,j}^2}\Big) \\ - \left(p_{x,i} \cdot p_{x,j} + p_{y,i} \cdot p_{y,j} + p_{z,i} \cdot p_{z,j}\right). \end{aligned} \tag{10}$$

Then, those detector hits left by the μ's from the $z \to \mu\mu$ decay is labeled as ground-truth interpretation. As our focus is on model interpretation performance, we only keep 10 tracks in each sample to train and test models to reduce classification difficulty, while raw files with all tracks are also provided. Each point in a sample is a detector hit left by some particle, and it is associated with a 3D coordinate. As points in `ActsTrack` do not have any features in $\mathbf{X}$, we use a dummy $\mathbf{X}$ with all ones when training models. Momentum of particles measured by the detectors is also provided, which can help evaluate fine-grained geometric patterns in the data, but it is not used for model training. Because each particle on average leaves 12 hits and a model may classify samples well if it captures the track of any one of the μ's from the decay, we report both precision@12 and precision@24. Finally, we randomly split the dataset into training/validation/test sets with a ratio of $70 : 15 : 15$.

Tau3Mu. The τ leptons produced in decays of D and B mesons simulated by the PYTHIA generator are used to generate the signal samples. The background events are generated with multiple soft QCD interactions modeled by the PYTHIA generator with a setting that resembles the collider environment at the High Luminosity LHC. The generated muons' interactions with the material in the endcap muon chambers are simulated including multiple scattering effects that resemble the CMS detector. The signal sample is mixed with the background samples at per-event level (per point cloud), with the ground-truth labels preserved for the interpretation studies. The hits left by the μ's from the $\tau \to \mu\mu\mu$ decay are labeled as ground-truth interpretation. We only use hits on the first layer of detectors to train models and make sure every sample in the dataset has at least three detector hits. Each point in the sample contains measurements of a local bending angle and a 2D coordinate in the pseudorapidity-azimuth (η-ϕ) space. Because in the best case the model only needs to capture hits from each μ, we report precision@3. And because 80% of positive samples have less than 7 hits labeled as ground-truth interpretation, we also report precision@7. Finally, we randomly split the dataset into training/validation/test sets with a ratio of $70 : 15 : 15$.

SynMol. We utilize the molecules in ZINC [93], and follow [33, 34] to create synthetic properties based on the existence of certain functional groups. Specifically, if a molecule contains both the unbranched alkane and carbonyl, then we label it as a positive sample; otherwise it is labeled as

a negative sample. So, the atoms in branched alkanes and carbonyl are labeled as ground-truth interpretation. Instead of 2D representations of molecules, we associate each atom a 3D coordinate by generating a conformer for each molecule. To do this, we first add hydrogens to the molecule and apply the ETKDG method [94]. After that, the generated structures are cleaned up using the MMFF94 force field [95] with a maximum iteration of 1000, and the added hydrogens are removed once it is finished. Both ETKDG and MMFF94 are implemented using RDKit. Besides a 3D coordinate, each point in a sample also has a categorical feature indicating the atom type. Even though the two functional groups may only have five atoms in total, some molecules may contain multiple such functional groups. So, we report both precision@5 and precision@8 (80% of positive samples have less than 8 atoms labeled as ground-truth interpretation). Finally, we split the dataset into training/validation/test sets in a way that the number of molecules with or without either of these functional groups is uniformly distributed following [33] so that the dataset bias is minimized.

PLBind. We utilize protein-ligand complexes from PDBBind [96], which annotates binding affinities for a subset of complexes in the Protein Data Bank (PDB) [97]. In PDBBind, each protein-ligand pair is annotated with a dissociation constant K_d, which measures the binding affinity between a pair of protein and ligand. We use a threshold of 10 nM on K_d to obtain a binary classification task, and the model interpretability is studied on the protein-ligand pairs with high affinities. To augment negative data, during training, there is a 10% change of switching the ligand of a complex to a random ligand, and the new protein-ligand pair will be labeled as a negative sample, i.e., low affinity. The ground-truth interpretation labels consist of two parts. First, as shown in previous studies [98], using the part of the protein that is within 15Å of the ligand is enough to even learn to generate ligands that bind to a certain protein, so, we define the amino acids that are within 15Å of the ligand to be the binding site and label them as ground-truth interpretation. Second, we retrieve all atomic contacts (hydrogen bond and hydrophobic contact) for every protein-ligand pair from PDBsum [99] and label the corresponding amino acids in the protein as the ground-truth interpretation. Each amino acid in a protein is associated with a 3D coordinate, the amono acid type, solvent accessible surface area (SASA), and the B-factor. Each atom in a ligand is associated with a 3D coordinate, the atom type, and Gasteiger charges. Finally, we split the dataset into training/validation/test sets according to the year the complexes are discovered following [100].

D Details on Hyperparameter Tuning

Gradient-based methods have no hyperparameters to tune, while all other methods are tuned based on the validation prediction performance for a fair comparison, because tuning on interpretation performance is impossible in a real-world setting. All settings are trained with 5 different seeds and the average performance on the 5 seeds are reported.

Basic Settings. We use a batch size of 128 on all datasets, except on `Tau3Mu` we use a batch size of 256 due to its large dataset size. Adam [101] optimizer with a learning rate of 1.0×10^{-3} and a weight decay of 1.0×10^{-5} are used. `ActsTrack`, `SynMol`, and `PLBind` construct k-nn graphs with k being 5; `Tau3Mu` constructs the graph by drawing edges for any pair of points with a distance less than 1. For a fair comparison, all models will be trained with 300 epochs on `ActsTrack` and `SynMol` and will be trained with 100 epochs on `Tau3Mu` and `PLBind`. For post-hoc methods, a classifier will be first pretrained with the epochs mentioned above, and those methods will further work on the model from the epoch with the best validation classification AUC during pretraining.

LRI-Bernoulli. β is tuned from $\{1.0, 0.1, 0.01\}$ after normalizing the summation of the KL divergence by the total number of points. α is tuned from $\{0.5, 0.7\}$. Note that α should not be less than 0.5, because Bern(0.5) injects the most randomness from an information-theoretic perspective. For `ActsTrack` and `SynMol`, we either train LRI-Bernoulli 300 epochs from scratch or first pre-train f with 200 epochs and then train both f and g with 100 epochs. For `Tau3Mu` and `PLBind`, we either train it 100 epochs from scratch or first pre-train f by 50 epochs and then train both f and g by 50 epochs. The temperature in the Gumbel-softmax trick is not tuned and is set to 1.

Table 4: Generalization performance of LRI. Classification AUC on test sets are reported with mean±std for all backbones on the four datasets.

	ActsTrack			Tau3Mu			SynMol			PLBind		
	DGCNN	Point Transformer	EGNN	DGCNN	Point Transformer	EGNN	DGCNN	Point Transformer	EGNN	DGCNN	Point Transformer	EGNN
ERM	97.99 ± 0.38	96.75 ± 0.21	97.45 ± 0.52	87.38 ± 0.08	86.20 ± 0.13	86.45 ± 0.09	99.95 ± 0.03	98.14 ± 0.42	99.87 ± 0.05	80.17 ± 5.23	83.13 ± 1.19	86.80 ± 3.52
LRI-Bernoulli	98.11 ± 0.09	97.39 ± 0.27	98.52 ± 0.35	87.45 ± 0.06	86.44 ± 0.08	86.56 ± 0.11	99.82 ± 0.15	97.93 ± 0.46	99.81 ± 0.05	81.25 ± 2.01	86.83 ± 2.06	86.93 ± 3.91
LRI-Gaussian	98.17 ± 0.24	98.04 ± 0.54	98.85 ± 0.14	87.74 ± 0.15	86.03 ± 0.37	86.67 ± 0.08	99.98 ± 0.01	98.80 ± 0.31	99.88 ± 0.05	84.34 ± 5.32	86.71 ± 0.64	87.53 ± 0.95

Table 5: Generalization performance of LRI with distribution shifts. The column name d_1-d_2 denotes the models are validated and tested on samples with d_1 and d_2 tracks, respectively.

ActsTrack	10-10	15-20	20-30	25-40	30-50	35-60	40-70
ERM	96.33 ± 0.65	93.83 ± 0.34	91.14 ± 1.07	87.85 ± 1.12	85.96 ± 1.74	84.19 ± 1.54	82.87 ± 0.74
LRI-Bernoulli	97.05 ± 0.71	94.66 ± 0.75	93.08 ± 0.94	90.93 ± 0.76	89.95 ± 1.31	86.11 ± 1.43	86.95 ± 1.46
LRI-Gaussian	97.51 ± 0.76	95.69 ± 0.80	93.64 ± 1.39	91.42 ± 2.97	89.89 ± 1.15	87.45 ± 1.47	88.13 ± 0.63

LRI-Gaussian. β and the training time is tuned in the same way as for LRI-Bernoulli. Instead of directly tuning σ, we tune it by rescaling the magnitude of the geometric features to make sure the KL divergence is stable for optimization, because the magnitude of $\mathbf{r}$ may vary significantly across different datasets, e.g., extremely small or large. Specifically, we rescale $\mathbf{r}$ by multiplying a constant c on it so that 1 would be roughly the 5th or 10th percentile of the entries in $\text{abs}(c \cdot \mathbf{r})$. In this way, c is tuned from $\{200, 300\}$ for ActsTrack, $\{10, 15\}$ for SynMol, $\{7, 11\}$ for Tau3Mu, and $\{0.9, 1.2\}$ for PLBind.

BernMask. The size loss coefficient and the mask entropy coefficient are both tuned from $\{1.0, 0.1, 0.01\}$. After the pretraining of the classifier, the number of mask update iterations per sample is set to 500 and the learning rate for each iteration is tuned from $\{0.1, 0.001\}$.

BernMask-P. After the pretraining of the classifier, the explainer module in BernMask-P will be trained with extra 100 epochs on ActsTrack and SynMol and with extra 50 epochs on Tau3Mu and PLBind. The size loss coefficient and the mask entropy coefficient are tuned in the same way as for BernMask. The temperature in the Gumbel-softmax trick is set to 1.

PointMask. It is trained with 300 epochs on ActsTrack and SynMol, and with 100 epochs on Tau3Mu and PLBind. α is tuned from $\{1.0, 0.1, 0.01\}$ and t is tuned from $\{0.2, 0.5, 0.8\}$.

E Implementation Details

Backbone Models. All backbone models have 4 layers with a hidden size of 64. Batch normalization [102] and ReLU activation [103] are used. All MLP layers use a dropout [104] ratio of 0.2. SUM pooling is used to generate point cloud embeddings to make predictions. All backbone models utilize implementations available in Pytorch-Geometric (PyG) [105].

LRI-Bernoulli. Directly removing points from $\mathcal{C}$ to yield $\tilde{\mathcal{C}}$ makes it non-differentiable. We provide two differentiable ways to approximate this step. The first way is to use another MLP to map the raw point features $\mathbf{X}$ to a latent feature space $\mathbf{H}$, and yield $\tilde{\mathcal{C}} = (\mathcal{V}, \mathbf{m}\mathbf{1}^T \odot \mathbf{H}, \mathbf{r})$. Here $\mathbf{m}$ is a vector containing m_v for each point $v \in \mathcal{V}$, $\mathbf{1}$ is a vector of all ones, and $\odot$ denotes element-wise product. This is because masking on $\mathbf{X}$ or $\mathbf{r}$ is inappropriate as values in them have specific physical meanings. When the backbone model is implemented using a message passing scheme, e.g., using PyG, another possible way is to directly use $\mathbf{m}$ to mask the message sent by the points to be removed. We find both ways can work well and we adopt the second way in our experiments.

LRI-Gaussian. We use a softplus layer to output a_1 and a_2 to parameterize the Gaussian distribution. To make it numerically stable we clip the results of the softplus layer to $[1.0 \times 10^{-6}, 1.0 \times 10^{6}]$. For ϕ in the differentiable graph reconstruction module, we find it empirically a simple MLP can work well enough and thus we adopt it to implement ϕ.

Baseline Methods. BernMask is extended from [58] based the authors' code and the implementation available in PyG. BernMask-P is extended from [59] based on the authors' code and a recent PR in PyG. PointMask is reproduced based on the authors' code. GradGAM and GradGeo are extended based on the code from [106].

Table 6: Ablation studies on the effect of the differentiable graph reconstruction module in LRI-Gaussian on ActsTrack and SynMol.

ActsTrack	DGCNN			Point Transformer			EGNN		
	ROC AUC	Prec@12	Prec@24	ROC AUC	Prec@12	Prec@24	ROC AUC	Prec@12	Prec@24
w/ Graph Recons.	94.74 ± 0.48	91.01 ± 1.02	78.47 ± 0.83	91.89 ± 2.07	88.05 ± 2.74	74.09 ± 3.61	94.43 ± 0.75	90.85 ± 0.69	76.42 ± 0.80
w/o Graph Recons.	67.03 ± 1.82	56.09 ± 2.11	41.52 ± 0.79	60.64 ± 1.93	39.16 ± 2.67	32.34 ± 1.42	82.50 ± 0.54	78.71 ± 3.84	59.02 ± 1.37

SynMol	DGCNN			Point Transformer			EGNN		
	ROC AUC	Prec@5	Prec@8	ROC AUC	Prec@5	Prec@8	ROC AUC	Prec@5	Prec@8
w/ Graph Recons.	98.83 ± 0.63	95.91 ± 1.73	76.86 ± 0.62	95.81 ± 1.42	89.61 ± 3.30	72.84 ± 2.12	96.88 ± 1.25	89.17 ± 5.03	74.47 ± 1.05
w/o Graph Recons.	98.65 ± 0.39	95.98 ± 1.43	76.58 ± 0.33	90.06 ± 1.51	79.80 ± 0.80	64.25 ± 1.31	78.39 ± 36.27	66.26 ± 35.52	58.79 ± 30.07

Discussions on LRI. We note that both f and g in LRI needs a permutation equivariant encoder to learn point representations, and we find for simple tasks these two encoders can share parameters to reduce model size without degrading interpretation performance, while for challenging tasks using two different encoders is beneficial. In out experiments we use the same encoder for ActsTrack, Tau3Mu, and SynMol, and use two different encoders for PLBind. If the model size is not a concern, using two encoders is generally recommended. The other thing is that one of the key components in LRI is the perturbation function h, as shown in Fig. 2, and we have shown two ways to design h, i.e., Bernoulli perturbation and Gaussian perturbation. Nonetheless, h can be generalized in many different ways. For example, h is where one can incorporate domain knowledge and provide contextualized interpretation results, i.e., human understandable results. For instance, instead of perturbing molecules in the atom level, it is possible to perturb molecules in the functional group level, e.g., by averaging the learned perturbation in each functional group, so that the interpretation results can be more contextualized and more human understandable. In this work, we experiment this feature on PLBind by replacing the learned $\{p_u\}_{u \in \mathcal{N}(v) \bigcup v}$ or $\{\mathbf{\Sigma}_u\}_{u \in \mathcal{N}(v) \bigcup v}$ in a neighbourhood with the minimum p or with the $\mathbf{\Sigma}$ having the maximum determinant in that neighbourhood, which encourages either aggressively perturbing all amino acids in a neighborhood or not perturbing any amino acids in the neighborhood, and we find this can better help discover binding sites for PLBind.

F Supplementary Experiments

F.1 Generalization Performance of LRI

LRI-induced models can generalize better while being interpretable. As shown in Table 4, both LRI-induced models never degrade prediction performance and sometimes may even boost it compared with models trained without LRI, i.e., using empirical risk minimization (ERM). Moreover, LRI-induced models are more robust to distribution shifts as LRI is grounded on the IB principle. Table 5 shows a study with shifts on the numbers of particle tracks, where all models are trained on samples with 10 particle tracks, and tested on samples with a different number of (from 10 to 70) particle tracks. We observe LRI-induced models work consistently better than models naively trained without LRI.

F.2 Ablation Studies

We also conduct ablation studies on the differentiable graph reconstruction module proposed specifically for geometric data in LRI-Gaussian, we find that without this module the interpretation performance of LRI-Gaussian may be reduced up to 49% on ActsTrack and up to 23% on SynMol, as shown in Table 6, which matches our expectations and validates the necessity of the proposed module. In this study, c is set to 200 on ActsTrack and to 5 on SynMol so that large perturbations are encouraged to better show the significance of this module, β is set to 0.01, and f is first trained by 200 epochs and then f and g are trained together by 100 epochs.

F.3 Fine-grained Geometric Pattern Learning

To discover fine-grained geometric information in ActsTrack using LRI-Gaussian as shown in Table 2, we conduct experiments based on DGCNN, where f is first trained with 100 epochs, and

then g is further trained with 500 epochs while f is finetuned with a learning rate of $1e$-8. f and g use two different encoders, β is set to 10, c is 100, all dropout ratio is set to 0, and $\frac{\mathbf{r}_v}{\|\mathbf{r}_v\|}$ is used as point features. Finally, we let g directly output covariance matrix $\mathbf{\Sigma}_v \in \mathbb{R}^{2\times 2}$, i.e., in x-y space, to only perturb the first two dimensions of $\mathbf{r}$.