INTERPRETABLE GEOMETRIC DEEP LEARNING VIA LEARNABLE RANDOMNESS INJECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

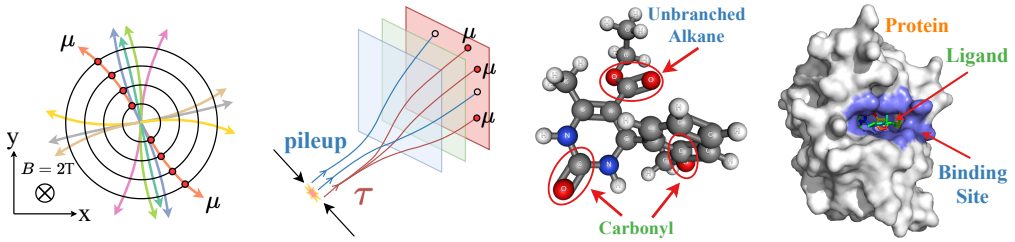
Point cloud data is ubiquitous in scientific fields. Recently, geometric deep learning (GDL) has been widely applied to solve prediction tasks with such data. However, GDL models are often complicated and hardly interpretable, which poses concerns to scientists when deploying these models in scientific 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. LRI-induced models, once being trained, can detect the points in the point cloud data that carry information indicative of the prediction label. We also propose four datasets from real scientific applications that cover the domains of high energy physics and biochemistry to evaluate the LRI mechanism. 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 is grounded by the information bottleneck principle, and thus LRI-induced models are also more robust to 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 (Guest et al., 2018); In biology, a protein is often measured and represented as a collection of amino acids with spacial locations (Wang et al., 2004; 2005). Geometric quantities of such point cloud data often encode important properties of the research object, analyzing which researchers may expect to achieve new scientific discoveries (Tusnady & Simon, 1998; Aad et al., 2012).

Recently, machine learning techniques have been employed to accelerate the procedure of scientific discovery (Butler et al., 2018; Carleo et al., 2019). For geometric data as above, geometric deep learning (GDL) (Bronstein et al., 2017; 2021) has shown great promise and has been applied to the fields such as HEP (Shlomi et al., 2020; Qu & Gouskos, 2020), biochemistry (Gainza et al., 2020; Townshend et al., 2021) 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 some symmetries of the system and incorporate the inductive biases reflected by geometric principles to guarantee their prediction quality (Cohen & Welling, 2016; Bogatskiy et al., 2020), and therefore often involve dedicated-designed complex NN architectures.

Albeit with outstanding prediction performance, the complication behind GDL models makes them hardly interpretable. However, in many scientific applications, interpretable models are in need (Roscher et al., 2020): 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 (Gao et al., 2018; Karimi et al., 2019; 2020). 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 (Nachman & Shimmin, 2019). Here,



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

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

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 Sec. 2. Moreover, recent works (Rudin, 2019; Laugel et al., 2019; Bordt et al., 2022; Miao et al., 2022) 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 (Miao et al., 2022), 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 (Tishby et al., 2000). 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 illustrate the four datasets in Fig. 1 and briefly introduce them below. More detailed descriptions can be found in Appendix B.

- `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 (Schulte et al., 2004; Thomson, 2013; Ai et al., 2022). *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 (Oerter, 2006; Blackstone et al., 2020), therefore, any detection of them is a clear signal of new physics beyond the Standard Model (Calibbi & Signorelli, 2018; Collaboration, 2021). 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 that explores the correlations between signal hits on top of background hits 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 (McCloskey et al., 2019; Sanchez-Lengeling et al., 2020), 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 (McCloskey et al., 2019) and use atoms in these functional groups 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 (Wang & Zhang, 2017; Karimi et al., 2019). 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 protein-ligand binding (Held et al., 2011; Du et al., 2016; Cang & Wei, 2018). *Our task* is to predict whether the affinity is above a given 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 DGCNN (Wang et al., 2019), Point Transformer (Zhao et al., 2021), and EGNN (Satorras et al., 2021) 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. Effective data patterns may vary regarding how the task 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 from the training to test scenarios.

2 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 (Zhou et al., 2016; Selvaraju et al., 2017; Sundararajan et al., 2017; Shrikumar et al., 2017; Chattopadhay et al., 2018) 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 (Ying et al., 2019; Luo et al., 2020; Schlichtkrull et al., 2021; Yuan et al., 2021). 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, Chen et al. (2018); Yoon et al. (2018) study pattern selection for regular data, Ribeiro et al. (2016); Lundberg & Lee (2017); Huang et al. (2022) utilize a local surrogate model, and Lundberg & Lee (2017); Chen et al. (2019b); Ancona et al. (2019); Lundberg et al. (2020) 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 (Bahdanau et al., 2015; Vaswani et al., 2017) were widely used for inherent interpretability, multiple recent studies show that they cannot provide reliable interpretation, especially for data with irregular structures (Serrano & Smith, 2019; Jain & Wallace, 2019; Ying et al., 2019; Luo et al., 2020). So, some works focusing on improving the attention mechanism for better interpretability (Bai et al., 2021; Miao et al., 2022), some propose to identify representative prototypes during training (Li et al., 2018; Chen et al., 2019a), and some methods (Taghanaki et al., 2020; Yu et al., 2021; Sun et al., 2022) adopt the

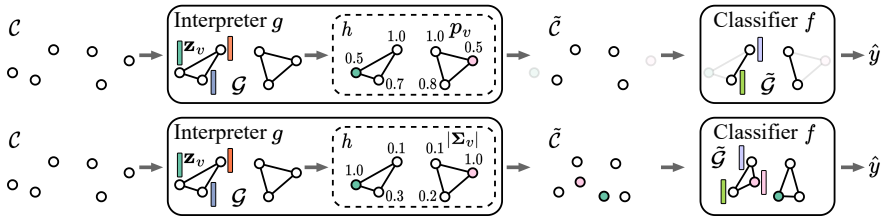


Figure 2: The architectures of LRI-Bernoulli (top) and LRI-Gaussian (bottom).

information bottleneck principle (Tishby et al., 2000). However, all these methods cannot analyze geometric features in GDL. Along another line, although invariant learning methods (Arjovsky et al., 2019; Chang et al., 2020; Krueger et al., 2021; Wu et al., 2022) 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 (Miao et al., 2022).

3 Preliminaries and Problem Formulation

In this section, we define some useful concepts and notations.

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 data (Zaheer et al., 2017; Charles et al., 2017). 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) (Kipf & Welling, 2017; Xu et al., 2019; Veličković et al., 2018) 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 (Fuchs et al., 2020; Gasteiger et al., 2020; Beaini et al., 2021; Satorras et al., 2021; Schütt et al., 2021). 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 (Schütt et al., 2017; Thomas et al., 2018; Wu et al., 2019). The *third* class will dynamically construct the k-nn graphs based on the hidden representations of points (Wang et al., 2019; Qu & Gouskos, 2020; Zhao et al., 2021). 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.

4 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. Then, we introduce LRI-Gaussian, which injects Gaussian randomness into geometric features to test the *location importance* of points. Finally, we connect our objectives with the information bottleneck principle (Tishby et al., 2000).

4.1 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 (Jang et al., 2017; Maddison et al., 2017) 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.

4.2 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 $\mathbf{\Sigma}_v \in \mathbb{R}^{3\times 3}$ 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.

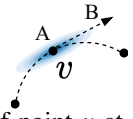
Table 1: Statistics of the four datasets.

	# 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

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 $\mathbf{U}_v \in \mathbb{R}^{3\times 3}$ 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 $\mathrm{AGG}(\{w_{vu}\mathbf{z}_u \mid u \in \mathcal{N}(v)\})$, 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.

4.3 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 (Tishby et al., 2000; Alemi et al., 2017) 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 Appendix A. 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 (Achille & Soatto, 2018; Wu et al., 2020; Miao et al., 2022).

5 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 B, C, and D, respectively. A summary of dataset statistics is shown in Table 1.

Backbone Models include DGCNN (Wang et al., 2019), Point Transformer (Zhao et al., 2021), and EGNN (Satorras et al., 2021) which have been widely used for scientific GDL (Qu & Gouskos, 2020; Atz et al., 2021; Gagliardi et al., 2022).

Baseline interpretation methods include three masking-based methods BernMask, BernMask-P and PointMask, and two gradient-based methods GradGeo and GradGAM. Masking-based methods

Table 2: 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	$\mathbf{79.44} \pm 2.62$	78.38 ± 1.96	52.55 ± 3.38
LRI-Bernoulli	$\mathbf{90.29} \pm 1.39$	$\mathbf{89.36} \pm 1.20$	$\mathbf{74.59} \pm 1.28$	$\mathbf{87.06} \pm 2.49$	$\mathbf{85.71} \pm 0.99$	$\mathbf{67.65} \pm 1.22$	78.57 ± 3.34	$\mathbf{81.56} \pm 1.13$	$\mathbf{55.29} \pm 1.99$
LRI-Gaussian	$\mathbf{95.49}^\dagger \pm 0.34$	$\mathbf{92.40}^\dagger \pm 0.64$	$\mathbf{79.87}^\dagger \pm 0.72$	$\mathbf{93.11}^\dagger \pm 1.50$	$\mathbf{88.39}^\dagger \pm 4.36$	$\mathbf{74.71}^\dagger \pm 1.99$	$\mathbf{94.34}^\dagger \pm 0.70$	$\mathbf{91.54}^\dagger \pm 1.49$	$\mathbf{76.78}^\dagger \pm 0.81$

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	$\mathbf{80.76} \pm 0.31$	68.86 ± 0.37	$\mathbf{56.05} \pm 0.34$	$\mathbf{79.62} \pm 0.33$	$\mathbf{67.96} \pm 0.49$	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	$\mathbf{80.91}^\dagger \pm 0.35$	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	$\mathbf{70.66} \pm 0.90$	55.94 ± 0.77	77.72 ± 1.52	67.73 ± 2.59	$\mathbf{55.74} \pm 1.15$	$\mathbf{78.71} \pm 0.66$	$\mathbf{65.99} \pm 0.84$	$\mathbf{55.98} \pm 0.57$
LRI-Gaussian	$\mathbf{81.38}^\dagger \pm 0.62$	$\mathbf{73.13}^\dagger \pm 1.10$	$\mathbf{58.28}^\dagger \pm 0.59$	79.58 ± 0.66	$\mathbf{70.32}^\dagger \pm 0.76$	$\mathbf{57.05}^\dagger \pm 0.53$	$\mathbf{80.02}^\dagger \pm 0.39$	$\mathbf{71.20}^\dagger \pm 0.93$	$\mathbf{57.07} \pm 0.41$

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	$\mathbf{85.52} \pm 5.75$	$\mathbf{68.94} \pm 6.37$
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	$\mathbf{96.03} \pm 1.54$	$\mathbf{87.11} \pm 4.51$	$\mathbf{74.57} \pm 1.57$	$\mathbf{91.69} \pm 1.52$	$\mathbf{82.72} \pm 2.20$	$\mathbf{68.37} \pm 1.11$	$\mathbf{90.64} \pm 3.30$	71.96 ± 5.97	68.08 ± 4.18
LRI-Gaussian	$\mathbf{99.02}^\dagger \pm 0.36$	$\mathbf{97.72}^\dagger \pm 0.94$	$\mathbf{77.04}^\dagger \pm 0.43$	$\mathbf{95.35}^\dagger \pm 1.02$	$\mathbf{87.09}^\dagger \pm 1.97$	$\mathbf{72.26}^\dagger \pm 1.40$	$\mathbf{97.28}^\dagger \pm 0.65$	$\mathbf{91.52}^\dagger \pm 1.28$	$\mathbf{74.05}^\dagger \pm 1.18$

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	$\mathbf{57.78}^\dagger \pm 2.61$	61.00 ± 2.24	60.11 ± 1.76
BernMask	48.18 ± 4.14	48.36 ± 3.32	48.00 ± 3.40	$\mathbf{59.73}^\dagger \pm 2.33$	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	$\mathbf{53.76} \pm 3.38$	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	$\mathbf{55.47}^\dagger \pm 2.06$	$\mathbf{67.56}^\dagger \pm 5.38$	$\mathbf{61.02}^\dagger \pm 5.68$	$\mathbf{59.53} \pm 1.94$	$\mathbf{72.98}^\dagger \pm 3.85$	$\mathbf{67.33}^\dagger \pm 2.08$	$\mathbf{57.07} \pm 3.09$	$\mathbf{73.22}^\dagger \pm 2.32$	$\mathbf{66.89} \pm 2.07$
LRI-Gaussian	51.81 ± 3.24	$\mathbf{63.88} \pm 3.18$	$\mathbf{60.37} \pm 3.09$	54.05 ± 2.94	$\mathbf{62.76} \pm 5.93$	$\mathbf{60.44} \pm 5.62$	50.32 ± 3.80	$\mathbf{72.64} \pm 2.04$	$\mathbf{69.40}^\dagger \pm 1.44$

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, i.e., from Ying et al. (2019) and Luo et al. (2020), 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 Taghanaki et al. (2020). Thus, they are the baselines of LRI-Bernoulli. GradGeo is extended from Shrikumar et al. (2017) 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 Selvaraju et al. (2017) 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. Then, 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. 5.5, 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.

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. For LRI-Bernoulli, α is tuned in $\{0.5, 0.7\}$; for LRI-Gaussian, as the geometric features $\mathbf{r}$ in our datasets are all centered at the origin, σ is tuned based on the 5th and 10th percentiles of entries of $\text{abs}(\mathbf{r})$. And, β is tuned in $\{1.0, 0.1, 0.01\}$ for both methods after normalization based on the total number of points.

Table 3: The angle (°, mean±std) between the velocity and the first principal component of $\mathbf{\Sigma}_v$ (LRI-Gaussian) v.s. between the velocity and the gradient of $\mathbf{r}_v$ in x-y space (GradGeo) 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

5.1 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 field $B = 2$T parallel to the z axis (see Fig. 1a). Table 2 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 Miao et al. (2022). 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 3 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 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 law – 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. Detailed descriptions and examples and can be found in Appendix G.

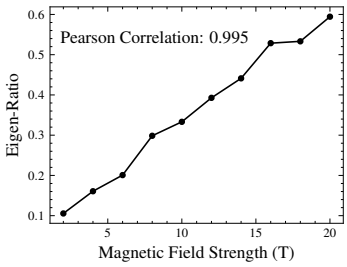


Figure 3: Eigen-ratio of $\mathbf{\Sigma}_v$ v.s. magnetic field strength B (T).

5.2 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 2, 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.

5.3 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 (McCloskey et al., 2019; Sanchez-Lengeling et al., 2020). 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 2, 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

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

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.

5.4 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 an 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 2, 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.

5.5 Generalization Performance and Ablation Studies 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. 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, which shows the significance of this module. More details of the ablation study can be found in Appendix E.1.

6 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

Georges Aad, Tatevik Abajyan, B Abbott, J Abdallah, S Abdel Khalek, Ahmed Ali Abdelalim, R Aben, B Abi, M Abolins, OS 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*, 2012.

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 2018.

Xiaocong Ai, Corentin Allaire, Noemi Calace, Angéla Czirkos, Markus Elsing, Irina Ene, Ralf Farkas, Louis-Guillaume Gagnon, Rocky Garg, Paul Gessinger, et al. A common tracking software project. *Computing and Software for Big Science*, 2022.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *International Conference on Learning Representations*, 2017.

Marco Ancona, Cengiz Oztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. *International Conference on Machine Learning*, 2019.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Kenneth Atz, Francesca Grisoni, and Gisbert Schneider. Geometric deep learning on molecular representations. *Nature Machine Intelligence*, 2021.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*, 2015.

Bing Bai, Jian Liang, Guanhua Zhang, Hao Li, Kun Bai, and Fei Wang. Why attentions may not be interpretable? *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. *International Conference on Machine Learning*, 2021.

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic Acids Research*, 2000.

Christian Bierlich, Smita Chakraborty, Nishita Desai, Leif Gellersen, Ilkka Helenius, Philip Ilten, Leif Lönnblad, Stephen Mrenna, Stefan Prestel, Christian T Preuss, Torbjorn Sjostrand, Peter Skands, Marius Utheim, and Rob Verheyen. A comprehensive guide to the physics and usage of pythia 8.3. *SciPost Physics*, 2022.

Patrick Blackstone, Matteo Fael, and Emilie Passemar. $\tau \to \mu\mu\mu$ at a rate of one out of 10^{14} tau decays? *The European Physical Journal C*, 2020.

Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. *International Conference on Machine Learning*, 2020.

Sebastian Bordt, Michèle Finck, Eric Raidl, and Ulrike von Luxburg. Post-hoc explanations fail to achieve their purpose in adversarial contexts. *ACM Conference on Fairness, Accountability, and Transparency*, 2022.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 2018.

Lorenzo Calibbi and Giovanni Signorelli. Charged lepton flavour violation: an experimental and theoretical introduction. *La Rivista del Nuovo Cimento*, 2018.

Zixuan Cang and Guo-Wei Wei. Integration of element specific persistent homology and machine learning for protein-ligand binding affinity prediction. *International Journal for Numerical Methods in Biomedical Engineering*, 2018.

Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 2019.

Shiyu Chang, Yang Zhang, Mo Yu, and Tommi Jaakkola. Invariant rationalization. *International Conference on Machine Learning*, 2020.

R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. *IEEE Winter Conference on Applications of Computer Vision*, 2018.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems*, 2019a.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. *International Conference on Machine Learning*, 2018.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data. *International Conference on Learning Representations*, 2019b.

Taco Cohen and Max Welling. Group equivariant convolutional networks. *International Conference on Machine Learning*, 2016.

Atlas Collaboration. Search for charged-lepton-flavour violation in z-boson decays with the atlas detector. *Nature Physics*, 2021.

Xing Du, Yi Li, Yuan-Ling Xia, Shi-Meng Ai, Jing Liang, Peng Sang, Xing-Lai Ji, and Shu-Qun Liu. Insights into protein–ligand interactions: mechanisms, models, and methods. *International Journal of Molecular Sciences*, 2016.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *International Conference on Learning Representations: Representation Learning on Graphs and Manifolds*, 2019.

Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 2020.

Luca Gagliardi, Andrea Raffo, Ulderico Fugacci, Silvia Biasotti, Walter Rocchia, Hao Huang, Boulbaba Ben Amor, Yi Fang, Yuanyuan Zhang, Xiao Wang, et al. Shrec 2022: Protein–ligand binding site recognition. *Computers & Graphics*, 2022.

Pablo Gainza, Freyr Sverrisson, Frederico Monti, Emanuele Rodola, D Boscaini, MM Bronstein, and BE Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 2020.

Kyle Yingkai Gao, Achille Fokoue, Heng Luo, Arun Iyengar, Sanjoy Dey, Ping Zhang, et al. Interpretable drug target prediction using deep neural representation. *International Joint Conference on Artificial Intelligence*, 2018.

Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *International Conference on Learning Representations*, 2020.

Jacob Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021.

Dan Guest, Kyle Cranmer, and Daniel Whiteson. Deep learning and its application to lhc physics. *Annual Review of Nuclear and Particle Science*, 2018.

Thomas A Halgren. Mmff vi. mmff94s option for energy minimization studies. *Journal of Computational Chemistry*, 1999.

Martin Held, Philipp Metzner, Jan-Hendrik Prinz, and Frank Noé. Mechanisms of protein-ligand association and its modulation by protein mutations. *Biophysical Journal*, 2011.

Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 2015.

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*, 2012.

Sarthak Jain and Byron C Wallace. Attention is not explanation. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations*, 2017.

Mostafa Karimi, Di Wu, Zhangyang Wang, and Yang Shen. Deepaffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks. *Bioinformatics*, 2019.

Mostafa Karimi, Di Wu, Zhangyang Wang, and Yang Shen. Explainable deep relational networks for predicting compound–protein affinities and contacts. *Journal of Chemical Information and Modeling*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). *International Conference on Machine Learning*, 2021.

Roman A Laskowski, Jagoda Jabłońska, Lukáš Pravda, Radka Svobodová Vařeková, and Janet M Thornton. Pdbsum: Structural summaries of pdb entries. *Protein Science*, 2018.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *International Joint Conference on Artificial Intelligence*, 2019.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. *AAAI Conference on Artificial Intelligence*, 2018.

Meng Liu, Youzhi Luo, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. Generating 3d molecules for target protein binding. *International Conference on Machine Learning*, 2022.

Zhihai Liu, Minyi Su, Li Han, Jie Liu, Qifan Yang, Yan Li, and Renxiao Wang. Forging the basis for developing protein–ligand interaction scoring functions. *Accounts of Chemical Research*, 2017.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 2017.

Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2020.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in Neural Information Processing Systems*, 2020.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations*, 2017.

Kevin McCloskey, Ankur Taly, Federico Monti, Michael P Brenner, and Lucy J Colwell. Using attribution to decode binding mechanism in neural network models for chemistry. *Proceedings of the National Academy of Sciences*, 2019.

Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. *International Conference on Machine Learning*, 2022.

Benjamin Nachman and Chase Shimmin. Ai safety for high energy physics. *arXiv preprint arXiv:1910.08606*, 2019.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. *International Conference on Machine Learning*, 2010.

Robert Oerter. *The theory of almost everything: The standard model, the unsung triumph of modern physics*. Penguin, 2006.

Huilin Qu and Loukas Gouskos. Jet tagging via particle clouds. *Physical Review D*, 2020.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. *ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2016.

Sereina Riniker and Gregory A Landrum. Better informed distance geometry: using what we know to improve conformation generation. *Journal of Chemical Information and Modeling*, 2015.

Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. Explainable machine learning for scientific insights and discoveries. *IEEE Access*, 2020.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 2019.

Benjamin Sanchez-Lengeling, Jennifer Wei, Brian Lee, Emily Reif, Peter Wang, Wesley Qian, Kevin McCloskey, Lucy Colwell, and Alexander Wiltschko. Evaluating attribution for graph neural networks. *Advances in Neural Information Processing Systems*, 2020.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. *International conference on machine learning*, 2021.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for {nlp} with differentiable edge masking. *International Conference on Learning Representations*, 2021.

Reinhard Schulte, Vladimir Bashkirov, Tianfang Li, Zhengrong Liang, Klaus Mueller, Jason Heimann, Leah R Johnson, Brian Keeney, HF-W Sadrozinski, Abraham Seiden, et al. Conceptual design of a proton computed tomography system for applications in proton radiation therapy. *IEEE Transactions on Nuclear Science*, 2004.

Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in Neural Information Processing Systems*, 2017.

Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. *International Conference on Machine Learning*, 2021.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *IEEE International Conference on Computer Vision*, 2017.

Sofia Serrano and Noah A Smith. Is attention interpretable? *Association for Computational Linguistics*, 2019.

Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2020.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *International Conference on Machine Learning*, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

Hannes Stärk, Octavian Ganea, Lagnajit Pattanaik, Dr.Regina Barzilay, and Tommi Jaakkola. EquiBind: Geometric deep learning for drug binding structure prediction. *International Conference on Machine Learning*, 2022.

Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and S Yu Philip. Graph structure learning with variational information bottleneck. *AAAI Conference on Artificial Intelligence*, 2022.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *International Conference on Machine Learning*, 2017.

Saeid Asgari Taghanaki, Kaveh Hassani, Pradeep Kumar Jayaraman, Amir Hosein Khasahmadi, and Tonya Custis. Pointmask: Towards interpretable and bias-resilient point cloud processing. *arXiv preprint arXiv:2007.04525*, 2020.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Mark Thomson. *Modern particle physics*. Cambridge University Press, 2013.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Raphael JL Townshend, Stephan Eismann, Andrew M Watkins, Ramya Rangan, Maria Karelina, Rhiju Das, and Ron O Dror. Geometric deep learning of rna structure. *Science*, 2021.

Gabor E Tusnady and Istvan Simon. Principles governing amino acid composition of integral membrane proteins: application to topology prediction. *Journal of Molecular Biology*, 1998.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.

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

Cheng Wang and Yingkai Zhang. Improving scoring-docking-screening powers of protein–ligand scoring functions using random forest. *Journal of Computational Chemistry*, 2017.

Renxiao Wang, Xueliang Fang, Yipin Lu, and Shaomeng Wang. The pdbbind database: Collection of binding affinities for protein- ligand complexes with known three-dimensional structures. *Journal of Medicinal Chemistry*, 2004.

Renxiao Wang, Xueliang Fang, Yipin Lu, Chao-Yie Yang, and Shaomeng Wang. The pdbbind database: methodologies and updates. *Journal of Medicinal Chemistry*, 2005.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions on Graphics*, 2019.

Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 2020.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Ying-Xin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. Discovering invariant rationales for graph neural networks. *International Conference on Learning Representations*, 2022.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *International Conference on Learning Representations*, 2019.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems*, 2019.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. Invase: Instance-wise variable selection using neural networks. *International Conference on Learning Representations*, 2018.

Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. *International Conference on Learning Representations*, 2021.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. *International Conference on Machine Learning*, 2021.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in Neural Information Processing Systems*, 2017.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. *IEEE International Conference on Computer Vision*, 2021.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

A Variational Bounds of the IB Principle

Let's assume the sample with its label $(\mathcal{C}, Y) \sim \mathbb{P}_{\mathbb{C}\times\mathbb{Y}}$. 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 $\tilde{\mathcal{C}}$ 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 the entropy of Y which is a constant. We use the prediction model f paired with the cross-entropy loss $\mathcal{L}_{CE}(f(\tilde{\mathcal{C}}), Y)$ to represent $-\mathbb{E}_{\tilde{\mathcal{C}},Y}\left[\log \mathbb{P}_\theta(Y \mid \tilde{\mathcal{C}})\right]$, minimizing which is thus equivalent to minimizing a variational upper bound of $-I(\tilde{\mathcal{C}};Y)$.

For the second term $I(\tilde{\mathcal{C}};\mathcal{C})$, because $\tilde{\mathcal{C}} = g(\mathcal{C})$. Suppose ϕ is the parameter of g. 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 $\mathbb{P}_\phi(\tilde{\mathcal{C}} \mid \mathcal{C}) = \prod_{v\in\mathcal{V}} \mathbb{P}(m_v|p_v)$, which essentially makes m_v conditionally independent across different points given the input point cloud $\mathcal{C}$. 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 $\mathbb{Q}(\tilde{\mathcal{C}}) = \sum_{\mathcal{C}} \mathbb{P}(\mathbf{m}' \mid \mathcal{C})\mathbb{P}(\mathcal{C})$. As $\mathbf{m}'$ is independent from $\mathcal{C}$ given its size n, $\mathbb{Q}(\tilde{\mathcal{C}}) = \sum_n \mathbb{P}(\mathbf{m}'|n)\mathbb{P}(\mathcal{C}=n) = \mathbb{P}(n)\prod_{v=1}^n \mathbb{P}(m'_v)$, 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 $c(n,\alpha)$ does not contain parameters to be optimized. 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 $\mathbb{P}_\phi(\tilde{\mathcal{C}} \mid \mathcal{C}) = \prod_{v\in\mathcal{V}} \mathbb{P}(\epsilon_v|\mathbf{\Sigma}_v)$, which essentially makes ϵ_v conditionally independent across different points given the input point cloud $\mathcal{C}$. 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.

B DETAILS OF THE DATASETS

B.1 BACKGROUND AND TASKS

We describe the background and tasks of our datasets in more detail in this section.

ActsTrack. As illustrated in Fig. 1a, protons will collide at the center of the detectors, and a number of interactions will happen, where different interactions may produce different particles, e.g., a $z \to \mu\mu$ decay produces two μ's. The produced particles will then fly through multiple layers of detectors with a magnetic field, and Fig. 1a shows an example that has four layers of ring detectors with a 2T magnetic field parallel to z-axis. When a particle flies through such detectors, the detectors will record where the particle hits the detector (i.e., geometric coordinates) and measure certain properties of the particle (e.g., momentum) depending on the type of the detector. And if the particle is charged, then its track in space may be curved due to the magnetic field. As such, each particle will leave a set of points on the detectors with some measured properties, and all particles produced from the collision will then form a point cloud, where a point is just a particle hit on a detector. However, we are only interested in a certain interaction, i.e., the $z \to \mu\mu$ decay, and do not care other interactions (these interactions are called pileup interactions). Fortunately, the particle hits in the point cloud will reveal the information about the interactions happened in the collision. For example, if a $z \to \mu\mu$ decay just happened, then there should be two sets of points in the point cloud that are left by the two μ's just produced from the decay. In principle, it is possible to reconstruct the type of a particle based on its tracks and/or some other measured properties in a magnetic field, and once we can find that there are two tracks that must be caused by two μ's (with certain invariant mass), then we would know a $z \to \mu\mu$ decay just happened in the collision. Therefore, the *classification* task of `ActsTrack` is to predict if there exists a $z \to \mu\mu$ decay or not in the collision given the point cloud measured by the detectors. Each positive sample contains particle hits from both a $z \to \mu\mu$ decay and some pileup interactions, while each negative sample has only hits from pileup interactions. In principle, if a classifier can successfully predict the existence of a $z \to \mu\mu$ decay, it should be aware of which sets of points in the point cloud represent the tracks of the μ's. Therefore, the particle hits left by the two μ's are labeled as ground-truth interpretation, and we expect an interpretable model to highlight these points as important points for the classification task.

Tau3Mu. Even though `Tau3Mu` has a similar basic HEP setup as `ActsTrack`, as shown in Fig. 1b, `Tau3Mu` now has a different configuration of the detectors and the magnetic field. Besides, the underlying physics process of interest is different and the task is with a different physics motivation (as shown in Sec. 1). The formulation of the machine learning task is a *classification* task to predict if there exists a $\tau \to \mu\mu\mu$ decay or not given a point cloud. Each positive sample contains both a $\tau \to \mu\mu\mu$ decay and some pileup interactions, while each negative sample has only pileup interactions. Likewise, if a classifier can successfully predict whether there is a $\tau \to \mu\mu\mu$ decay or not, we expect those particle hits left by the three μ's produced by the $\tau \to \mu\mu\mu$ decay to be the important points for the classification task, and an interpretable model should be able to highlight these points.

SynMol. Different from typical molecular property prediction tasks, `SynMol` provides 3D representations of molecules instead of 2D chemical bond graphs. Namely, each molecule is represented as a 3D point cloud with each point being an atom in space. The *classification* task of `SynMol` is to predict if a molecule has a certain property or not. As shown in Fig. 1c, we know that the property of interest is determined by two certain functional groups (i.e., specific groupings of atoms). Therefore, we expect an interpretable model that can successfully predict the property should be able to highlight those atoms in the two functional groups as the important points for the classification task.

PLBind. The *classification* task of `PLBind` is to predict if a pair of protein and ligand can bind together (with a high affinity) or not given the 3D structures of proteins and ligands. Notably, Fig. 1d only shows the surface of a protein for better visualization, and actually proteins in `PLBind` are represented as 3D point clouds, where each point is just an amino acid in the protein. Similarly, ligands are just small molecules, and they are also represented as 3D point clouds, where each point is an atom in the ligand. In principle, as shown in Fig. 1d, a high binding affinity should largely depend on the amino acids of the protein that are near the binding site. Therefore, we label those amino acids near the binding site as ground-truth interpretation, and we expect an interpretable classifier that can successfully predict the high-affinity pairs should be able to highlight those amino acids as important points for the classification task.

B.2 DATASET COLLECTION

We elaborate how we collect the four datasets in this section. The statistics of the four datasets are shown in Table 1.

Basic Settings. 1) For each sample in the four datasets, all points 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 PYTHIA generator (Bierlich et al., 2022) overlaid with soft QCD pileup events, and particle tracks are simulated using Acts Common Tracking Software (Ai et al., 2022). 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.

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

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. In the future works, we will study a more extensive evaluation of different approaches over the samples with all tracks. 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. Momenta of particles measured by the detectors are 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 (Bierlich et al., 2022) 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 (Irwin et al., 2012), and follow McCloskey et al. (2019) and Sanchez-Lengeling et al. (2020) 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 (Riniker & Landrum, 2015). After that, the generated structures are cleaned up using the MMFF94 force field (Halgren, 1999) 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 (McCloskey et al., 2019) so that the dataset bias is minimized.

PLBind. We utilize protein-ligand complexes from PDBBind (Liu et al., 2017), which annotates binding affinities for a subset of complexes in the Protein Data Bank (PDB) (Berman et al., 2000). 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 (Liu et al., 2022), 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 (Laskowski et al., 2018) 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 Stärk et al. (2022).

C Details on Hyperparameter Tuning

All hyperparameters are tuned based on validation classification AUC for a fair comparison. 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. The Adam (Kingma & Ba, 2015) 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`, so that all models are converged. 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 (also based on the best validation prediction performance). Similarly, 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.

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 keep $\sigma = 1$ and 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`.

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

BernMask. This approach generalizes GNNExplainer (Ying et al., 2019) and learns a node mask $\mathbf{m} \in [0,1]^n$ for each sample $\mathcal{C}$. Similar to GNNExplainer, there are two regularizers in its loss function, that is, mask size and entropy constraints. Specifically, the coefficient of the ℓ_1 penalty on the normalized mask size, i.e., $\frac{\|\mathbf{m}\|_1}{n}$, is tuned from $\{1.0, 0.1, 0.01\}$. Element-wise entropy of $\mathbf{m}$ is applied, i.e., $\frac{1}{n}\sum_i -m_i \log m_i - (1-m_i)\log(1-m_i)$ and m_i is the ith entry of $\mathbf{m}$, to encourage generating discrete masks, and the coefficient of this entropy constraint is tuned from $\{1.0, 0.1, 0.01\}$. After pretraining the classifier, we set the number of iterations to learn masks per sample as 500 and tune the learning rate for each iteration from $\{0.1, 0.001\}$.

BernMask-P. This approach generalizes PGExplainer (Luo et al., 2020) and learns a node mask $\mathbf{m} \in [0,1]^n$ for each sample $\mathcal{C}$. Similar to BernMask, it also has mask size and entropy constraints in its loss function, and their coefficients are both tuned from $\{1.0, 0.1, 0.01\}$ as well. After pretraining 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 temperature in the Gumbel-softmax trick is set to 1.

PointMask (Taghanaki et al., 2020)**.** It is trained with 300 epochs on `ActsTrack` and `SynMol`, and with 100 epochs on `Tau3Mu` and `PLBind`. It has two hyperparameters as specified in their paper, i.e., α and t, where α is tuned from $\{1.0, 0.1, 0.01\}$ and t is tuned from $\{0.2, 0.5, 0.8\}$.

D IMPLEMENTATION DETAILS

Backbone Models. All backbone models have 4 layers with a hidden size of 64. Batch normalization (Ioffe & Szegedy, 2015) and ReLU activation (Nair & Hinton, 2010) are used. All MLP layers use a dropout (Srivastava et al., 2014) 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) (Fey & Lenssen, 2019).

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 by using a message passing scheme, e.g., using PyG, another possible way is to 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 Ying et al. (2019) based the authors' code and the implementation available in PyG. BernMask-P is extended from Luo et al. (2020) 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 Gildenblat & contributors (2021).

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 may be beneficial. In our 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 can be a good starting point. 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.

E Supplementary Experiments

E.1 Ablation Studies

Table 6 shows the effect of the differentiable graph reconstruction module in LRI-Gaussian, where 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. We observe significant drops when LRI-Gaussian is trained without differentiable graph reconstruction, which matches our expectations and validates the necessity of the proposed module.

E.2 Fine-grained Geometric Pattern Learning

To discover fine-grained geometric information in ActsTrack using LRI-Gaussian as shown in Table 3, 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 $\mathbf{r}$.

F Limitations and Future Directions of LRI

- Current LRI provides interpretation results at point-level, while cannot find out geometric patterns shared by a group of points (e.g., some set of points may be allowed to rotate around a reference point in space).
- Although Gaussian noise can help capture some fine-grained geometric patterns, if the underlying geometric patterns get too complicated, LRI-Gaussian may degrade to only offer a rank of point importance and have limited capability to capture meaningful fine-grained geometric patterns. It may be possible to design customized noise with prior knowledge to detect different types of geometric patterns (e.g., what if some points can be freely moved along a curve without affecting prediction loss?), and can we even not use a fixed noise distribution (i.e., Gaussian or Bernoulli) and also make it trainable?
- Current LRI may provide interpretation results that are not contextualized enough to guide scientists for further research. For example, chemists may prefer an interpretable model to tell them directly which functional groups are important instead of which atoms are important.

G Interpretation Visualization

ActsTrack. Figure 4 shows the interpretable patterns discovered by LRI-Gaussian. The top row shows that the randomness of the learned Gaussian noise $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_v)$ indicates the importance of

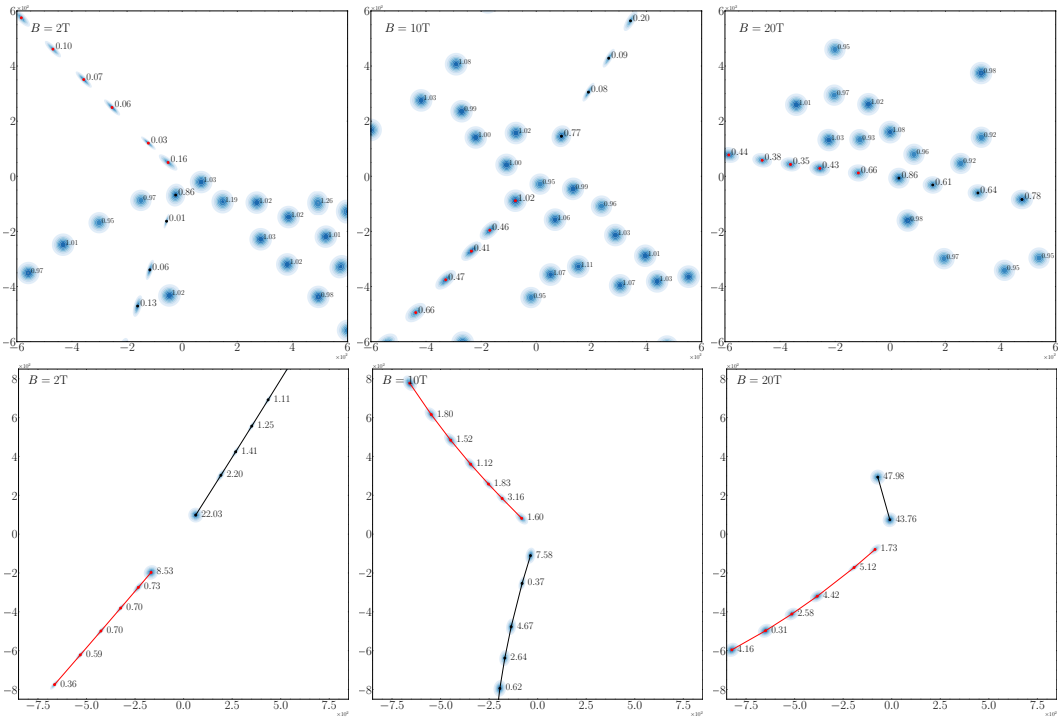


Figure 4: Visualizing interpretable patterns discovered by LRI-Gaussian for `ActsTrack`, where points are plotted in x-y plane (mm) with magnetic fields perpendicular to the plane, only points colored black/red indicate the two tracks of the two μ's from the $z \to \mu\mu$ decay, and the blue contour superimposed on each point v is the shape of the (scaled) learned Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}'_v)$, where $\mathbf{\Sigma}'_{v_{i,j}} = 400\mathbf{\Sigma}_{v_{i,j}}$, i.e., each raw element is multiplied by 400, as otherwise the contour would be hardly visible. For figures in the top row, the number next to each point v is $|\mathbf{\Sigma}_v|$; for figures in the bottom row, the number next to each point is the angle between the true track direction and the learned direction, and non-critical points are not plotted for better visualization.

each point v, and thus $|\mathbf{\Sigma}_v|$ can measure the importance quantitatively as it characterizes the entropy of Gaussian. In the examples, non-critical points learn isotropic Gaussian close to $\mathcal{N}(\mathbf{0}, \sigma\mathbf{I})$ as regularized by the KL divergence in Eq. 2, while critical points left by the μ's produced by the $z \to \mu\mu$ decay learn multivariate Gaussian with small randomness, i.e., small $|\mathbf{\Sigma}_v|$, because high randomness on these points would greatly affect model prediction and the prediction loss in Eq. 2 would not allow this to happen. As shown quantitatively in Table 2, $|\mathbf{\Sigma}_v|$ measures the importance of each point very well and can provide the best interpretation AUC in most cases.

The bottom row of Figure 4 shows that LRI-Gaussian is also capable of discovering fine-grained geometric patterns. In the examples, small perturbation of each point along the underlying track direction may not even affect the model prediction (because the track radius may not change a lot by such perturbations). Therefore, the principal component (i.e., the eigenvector that corresponds to the largest eigenvalue) of $\mathbf{\Sigma}_v$ in x-y space can give an estimation of the track direction at v. As shown quantitatively in Table 3, the principal component can actually estimate the track direction very well. Meanwhile, we can also see that the ratio between the lengths of two principal components (i.e., $\frac{2^{\text{nd}} \text{ Largest Eigenvalue}}{\text{Largest Eigenvalue}}$) of $\mathbf{\Sigma}_v$ varies when the strength of the background magnetic field B changes, and as shown quantitatively in Figure 3, this ratio provides an almost accurate estimation of B up to a constant multiplier. This is because the direction orthogonal to the track direction is the direction of Lorentz force, and when the principal component of $\mathbf{\Sigma}_v$ in x-y space indicates the track direction, the second principal component in x-y space would be the direction of Lorentz force. Since Lorentz force $F \propto B$ and the properties of μ's (e.g., momentum) from the $z \to \mu\mu$ decay is independent of magnetic field strength, the ratio $\frac{2^{\text{nd}} \text{ Largest Eigenvalue}}{\text{Largest Eigenvalue}}$ can imply the strength of the background magnetic field.