Ising on the Graph: Task-specific Graph Subsampling via the Ising Model

Maria Bånkestad
Uppsala University
RISE
maria.bankestad@ri.se

Jennifer R. Andersson
Uppsala University

Sebastian Mair
Linköping University
Uppsala University

Jens Sjölund
Uppsala University

Abstract

Reducing a graph while preserving its overall properties is an important problem with many applications. Typically, reduction approaches either remove edges (sparsification) or merge nodes (coarsening) in an unsupervised way with no specific downstream task in mind. In this paper, we present an approach for subsampling graph structures using an Ising model defined on either the nodes or edges and learning the external magnetic field of the Ising model using a graph neural network. Our approach is task-specific as it can learn how to reduce a graph for a specific downstream task in an end-to-end fashion, without requiring a differentiable loss function for the task. We showcase the versatility of our approach on four distinct applications: image segmentation, explainability for graph classification, 3D shape sparsification, and sparse approximate matrix inverse determination.

1 Introduction

Hierarchical organization is a recurring theme in nature and human endeavors [1]. Part of its appeal lies in its interpretability and computational efficiency, especially on uniformly discretized domains such as images or time series [2–5]. On graph-structured data, it is, however, no longer as clear what it means to, e.g., *"sample every second point"*. Solid theoretical arguments favor a spectral approach [6], wherein a simplified graph is found by maximizing the spectral similarity to the original graph [7]. On the other hand, instead of retaining as much of the original information as possible, it may often be more valuable to distill the critical information for the task at hand [8, 9]. The two main approaches for graph simplification, coarsening, and sparsification, involve removing nodes and edges. However, it is impossible to tailor coarsening or sparsification to a particular task using existing methods.

We take inspiration from the well-known Ising model in physics [10], which emanated as an analytical tool for studying magnetism but has since undergone many extensions [11, 12]. Within the Ising model, each location is associated with a binary state (interpreted as pointing up or down), and the configuration of all states is associated with an energy. Thus, the Ising model can be seen as an energy-based model with an energy function comprising a pairwise term and a pointwise "bias" term. As illustrated in Figure 1, the sign of the pairwise term determines whether neighboring states attract or repel each other, while the pointwise term (corresponding to the local magnetic field) controls the propensity of a particular alignment.

In this paper, we consider the Ising model defined on a graph's nodes or edges and augment it with a graph neural network that models the local magnetic field. There are no particular restrictions on the type of graph neural network, which means that it can, e.g., process multidimensional node and edge features. Since a node (or edge) is either included or not, we use techniques for gradient estimation in discrete models to train our model for a given task. Specifically, we show that the two-point version of the REINFORCE Leave-One-Out estimator [13] allows non-differentiable task-specific losses. We demonstrate the broad applicability of our model through examples from four domains: image segmentation, explainability for graph classification, 3D shape sparsification, and linear algebra.

Maria Bånkestad, Ising on the Graph: Task-specific Graph Subsampling via the Ising Model. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

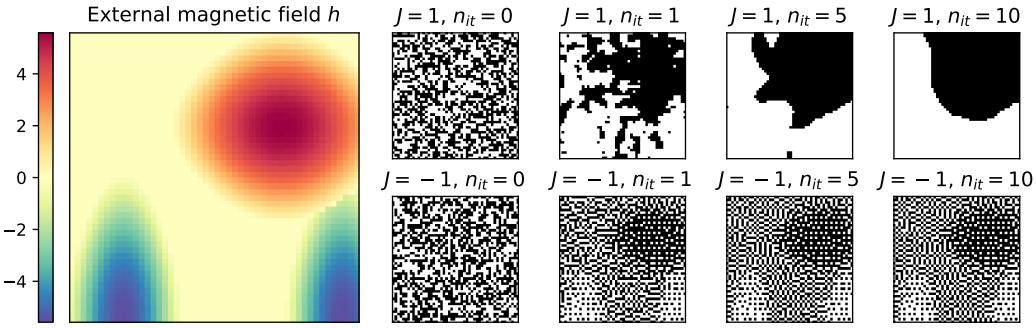

Figure 1: The external magnetic field h (left) can vary spatially and influences the sampling probability relative to its strength. The sign of the coupling constant J determines whether neighboring spins attract (top right) or repel each other (bottom right).

2 Background

Energy-based Models. Energy-based models (EBMs) are defined by an energy function $E_\theta \colon \mathbb{R}^d \to \mathbb{R}$, parameterized by θ, where d represents the dimensionality of the input space. The density defined by an EBM is given by the Boltzmann distribution [14]

$$p_\theta(x) = Z_\theta^{-1} \exp(-\beta E_\theta(x)), \tag{1}$$

where $\beta > 0$ is the inverse temperature and the normalization constant $Z = \int \exp(-\beta E_\theta(x)) \mathrm{d}x$, also known as the partition function, is typically intractable.

A sample with low energy will have a higher probability than a sample with high energy. Many approaches have been proposed for training energy-based models [15], but most of them target the generative setting where a training dataset is given and training amounts to (approximate) maximum likelihood estimation.

Ising Model. The Ising model corresponds to a deceptively simple energy-based model that is straightforward to express on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of a tuple of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ connecting the nodes. Specifically, the Ising energy is of the form [12]

$$E(x) = -\sum_{(i,j)\in\mathcal{E}} J_{ij} x_i x_j - \sum_{i \in \mathcal{V}} h_i x_i, \tag{2}$$

where the nodes (indices i, j) are associated with spins $x_i \in \{\pm 1\}$, the interactions $J_{ij} \in \mathbb{R}$ determine whether the behavior is ferromagnetic ($J > 0$) or antiferromagnetic ($J < 0$), and h_i is the external magnetic field.

It follows from Equation (2) that in the absence of an external magnetic field, neighboring spins strive to be parallel in the ferromagnetic case and antiparallel in the antiferromagnetic case. An external magnetic field that can vary across nodes influences the sampling probability of the corresponding nodes. The system is disordered at high temperatures (small β) and ordered at low temperatures.

Graph Neural Network. A graph neural network (GNN) consists of multiple message-passing layers [16]. Given a node feature x_i^k at node i and edge features e_{ij}^k between node i and its neighbors $\{j \colon j \in \mathcal{N}(i)\}$, the message passing procedure at layer k is defined as

$$m_{ij}^k = f^{\mathrm{m}}(x_i^k, x_j^k, e_{ij}^k), \quad \hat{x}_i^{k+1} = f^{\mathrm{a}}_{j\in\mathcal{N}(i)}(m_{ij}^k), \quad x_i^{k+1} = f^{\mathrm{u}}(x_i^k, \hat{x}_i^{k+1}), \tag{3}$$

where f^{m} is the message function, deriving the message from node j to node i, and $f^{\mathrm{a}}_{j\in\mathcal{N}(i)}$ is a function that aggregates the messages from the neighbors of node i, denoted $\mathcal{N}(i)$. The aggregation function f^{a} is often just a simple sum or average. Finally, f^{u} is the update function that updates the features for each node. A GNN consists of message-passing layers stacked onto each other, where the node output from one layer is the input of the next layer.

Algorithm 1 Monte Carlo Sampling of the Ising Model

Input: Graph $\{\mathbf{G}_c\}_{c=1}^{C}$ grouped by color c, interactions term J, external magnetic field h, iterations T.
Output: State configuration $x \in \{\pm 1\}^{|\mathcal{V}|}$.
for $i = 1$ **to** $|\mathcal{V}|$ **do**
  $x_i \sim U(\{-1, 1\})$ {Sample an initial state}
for $t = 1$ **to** T **do**
  for $c = 1$ **to** C **do**
    for all nodes $x_i \in \mathbf{G}_c$ **do**
      $\Delta E_i = 2x_i \left(J \sum_{j \in \mathcal{N}(i)} x_j + h_i \right)$
      $r \sim U([0, 1])$ {Sample r uniformly at random}
      if $\Delta E_i < 0$ **or** $\exp(-2\beta \Delta E) > r$ **then**
        $x_i = -1 \cdot x_i$ {Flip the spin of node i}

Note that we can rewrite the energy in Equation (2) to reveal the message-passing structure of the Ising model as

$$E(x^k) = -\sum_{i \in \mathcal{V}} x_i^k \cdot \left(h_i + \sum_{j \in \mathcal{N}(i)} J_{ij} x_j^k \right) = -\sum_{i \in \mathcal{V}} x_i^k h_i^{\text{eff}}(x^k), \quad h_i^{\text{eff}}(x^k) = h_i + \sum_{j \in \mathcal{N}(i)} J_{ij} x_j^k, \tag{4}$$

where $h_i^{\text{eff}}(x^k)$ is the effective magnetic field at the k-th iteration. Equation (4) can be viewed as a message-passing step in a GNN, where the message function f^{m} is $m_{ij}^k = J_{ij} x_j^k$, the aggregation function f^{a} is the sum $\hat{x}_i^{k+1} = \sum_{j \in \mathcal{N}(i)} m_{ij}^k$, and the update function f^{u} is $x_i^{k+1} = -x_i^k \left(h_i + \hat{x}_i^{k+1} \right)$. Here, h_i is the external magnetic field at node i, while x_i^k is the spin state of node i at iteration k. The spin state updates based on h_i and the states of neighboring nodes $\{x_j^k\}_{j \in \mathcal{N}(i)}$.

Gradient Estimation in Discrete Models. Let $p_\theta(x)$ be a discrete probability distribution and assume that we want to estimate the gradient of a loss defined as the expectation of a loss $\ell(x)$ over this distribution, i.e.,

$$\mathcal{L}(\theta) = \mathbb{E}_{x \sim p_\theta(x)} \left[\ell(x) \right]. \tag{5}$$

Because of the discreteness, it is impossible to compute a gradient directly using backpropagation. Through a simple algebraic manipulation, we can rewrite this expression as

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)} \left[\ell(x) \right] = \mathbb{E}_{x \sim p_\theta(x)} \left[\ell(x) \nabla_\theta \log p_\theta(x) \right], \tag{6}$$

which is called the REINFORCE estimator [17]. Though general, it suffers from high variance. A way to reduce this variance is to use the REINFORCE Leave-One-Out (RLOO) estimator [13],

$$(\nabla_\theta \mathcal{L})_{\text{LOO}}^K = \frac{1}{K} \sum_{k=1}^{K} \left(\ell(x^{(k)}) - \bar{\ell}_j \right) \nabla_\theta \log p_\theta(x^{(k)}), \quad \text{where } \bar{\ell}_j = \frac{1}{K-1} \sum_{j=1, j \neq k}^{K} \ell(x^{(j)}). \tag{7}$$

3 Method

In GNN terminology, the interactions J_{ij} represent edge features, while the external magnetic fields h_i serve as node features. The spin state of each node x_i is a dynamic variable that can take values in $\{\pm 1\}$. In particular, we use a GNN to parameterize the external magnetic field, $h_\theta \colon \mathcal{G} \to \mathbb{R}^{|\mathcal{V}|}$, defining the Ising model according to Equation (1). The final output is a binary partitioning of the input graph, given by sampling this energy-based model.

Sampling. We use the Metropolis-Hastings algorithm [18] for the sampling, see Algorithm 1. However, to make it computationally efficient, we parallelize it by first coloring the graph so that nodes of the same color are never neighbors. Then, we can perform simultaneous Metropolis-Hastings updates for all nodes of the same color. A simple checkerboard pattern perfectly colors grid-structured graphs, such as images. For general graphs, we run a greedy graph coloring heuristic [19] that produces a coloring with a limited, but not necessarily minimal, number of colors C (see Appendix B).

Controlling the Sampling Fraction. For some tasks, the obvious way to minimize the loss is to either sample all nodes or none. To counteract this, we need a way to control the fraction of nodes to

sample or, equivalently, to control the average magnetization η, which is defined as

$$\eta = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \mathbb{E}_{x \sim p_\theta(x)} [x_i] , \tag{8}$$

where $x \in \{\pm 1\}^{|\mathcal{V}|}$ denotes the vector containing all states and x_i is the ith entry in x. Let $\bar{x}_i = \mathbb{E}_{x \sim p_\theta(x)} [x_i]$ denote the local magnetization at node i. Since the Markov blanket of a node is precisely its neighborhood, we can use Equation (4) to rewrite the local magnetization as

$$\bar{x}_i = \mathbb{E}_{j \in \mathcal{N}(i)} \left[\mathbb{E}_i [x_i \mid x_j] \right] = \mathbb{E}_{j \in \mathcal{N}(i)} \left[\tanh(\beta h_i^{\text{eff}}(x)) \right] . \tag{9}$$

The corresponding mean-field (variational) approximation is a nonlinear system of equations in $\bar{x}$ [18]:

$$\bar{h}_i^{\text{eff}}(\bar{x}) = h_i + \sum_{j \in \mathcal{N}(i)} J_{ij} \bar{x}_j, \quad \bar{x}_i = \tanh(\beta \bar{h}_i^{\text{eff}}). \tag{10}$$

Solving this yields a deterministic way of approximating the average magnetization η. However, we are primarily interested in the ordered regime, where $\beta \bar{h}_i^{\text{eff}}$ is large, and it approximately holds that $x_j = J_{ij} x_i$, which implies that $\bar{x}_i = \tanh(\beta h_i)$ and, thus,

$$\tilde{\eta}_{\text{det}} = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \tanh (\beta h_i) . \tag{11}$$

This can be contrasted with the stochastic estimate we get from using a one-sample Monte Carlo estimate in Equation (9):

$$\tilde{\eta}_{\text{sto}} = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \tanh \left(\beta h_i^{\text{eff}}(x) \right) , \quad x \sim p_\theta(x). \tag{12}$$

Learning. In contrast to most other energy-based models [15], ours is not an unconditional generative model. Neither is it a supervised regression task [20], since we *learn to subsample* without requiring ground truth labels on which nodes are the correct ones to choose. Instead, we target the scenario where it is challenging to specify useful subgraphs for a downstream task but we have access to an associated, possibly non-differentiable, loss (which may use labels for the downstream task, as in Section 4.2). However, supervised training is also possible, as we demonstrate in Section 4.1.

We need the gradient of the log probability for gradient-based training of the GNN in our Ising model, which can be decomposed as a sum of two terms:

$$\nabla_\theta \log p_\theta(x) = -\beta \nabla_\theta E_\theta(x) - \nabla_\theta \log Z_\theta. \tag{13}$$

We are particularly interested in training the model for a downstream task associated with a loss $\ell(x)$ defined in terms of samples $x \sim p_\theta(x)$ from the Ising model. The goal is then to minimize the expected loss as defined in Equation (5).

Since we are operating in a discrete setting, we use the gradient estimation technique described in Section 2. Specifically, using $K = 2$ in Equation (7), the problematic $\log Z_\theta$ terms cancel in the RLOO estimator, and we obtain

$$(\nabla_\theta \mathcal{L})^2_{\text{LOO}} = -\frac{\beta}{2} \left(\ell(x^{(1)}) - \ell(x^{(2)}) \right) \cdot \nabla_\theta \left(E_\theta(x^{(1)}) - E_\theta(x^{(2)}) \right) . \tag{14}$$

To train the model, we can thus draw two independent samples $x^{(1)}, x^{(2)} \sim p_\theta(x)$ from the Ising model and estimate the gradient using Equation (14), where $\nabla_\theta \left(E_\theta(x^{(1)}) - E_\theta(x^{(2)}) \right)$ can be computed by automatic differentiation.

To control the sampling fraction, as described in Section 3, we could either include a regularization term that depends on the stochastic sampling fraction (Equation (12)) directly in the task-specific loss ℓ, or use a deterministic approximation based on Equation (11) to move it outside the expectation and thereby enable direct automatic differentiation. Empirically, we found the latter option to work better. Specifically, we use the training loss

$$\mathcal{L}_{\text{tot}}(\theta) = \mathbb{E}_{x \sim p_\theta(x)} [\ell(x)] + (\tilde{\eta}_{\text{det}}(\theta) - \eta)^2, \tag{15}$$

where the desired value of the average magnetization η is defined as in Equation (8).

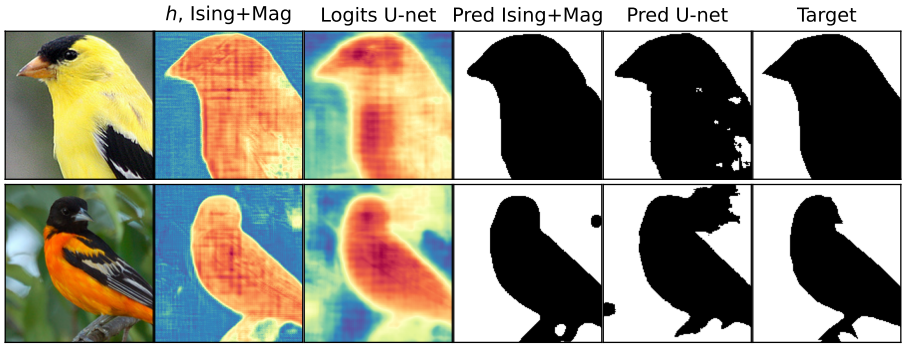

Figure 2: Comparison of a U-Net trained with a cross-entropy loss against an Ising model with a learned magnetic field. From left to right are the image, the Ising model magnetic field, the U-Net output (the logits), the Ising model prediction, the U-Net prediction, and the true segmentation mask.

4 Applications

We apply the proposed energy-based graph subsampling method across four distinct areas: image segmentation, explainability for graph classification, 3D shape sparsification, and sparse approximate matrix inverses. Code is available at `https://github.com/mariabankestad/IsingOnGraphs`.

4.1 Illustrative Example on Image Segmentation

To provide an intuitive visual demonstration, we first apply our approach to an image segmentation task, even though our main objective is learning Ising models on general graphs. An image can be represented as a graph, where pixels are nodes and edges denote neighboring pixel relationships [21]. In the ferromagnetic case ($J = 1$), the Ising model encourages nearby pixels to have similar values, resulting in smooth, coherent segmentations. Using a probabilistic graphical model to post-process neural network outputs can significantly improve object boundary localization [22]. Unlike the general setting in Section 3, this example has ground-truth labels (segmentation masks), allowing for a supervised approach with no downstream task involved.

Our Approach. Because of the grid structure, the message from the 8-connected neighbors can be implemented as a convolutional layer with kernel size three and zero padding. Specifically, we define the energy per pixel as:

$$E_i = -x_i \left(\mathrm{Conv}_W \left(x_i \right) + h_\theta(x_i) \right), \quad W = \begin{bmatrix} w^* & w & w^* \\ w & 0 & w \\ w^* & w & w^* \end{bmatrix}.$$

Here, Conv_W is an image convolutional operator with weights w, and $h_\theta(x_i)$ is an external magnetic field with learnable parameters θ. The weights w and w^* have the same sign but can have different values. In contrast to Zheng et al. [23], who use the mean-field approximation in Equation (10) to learn a fully-connected conditional random field end-to-end, our approach applies to graph-structured energy functions and allows task-specific losses.

Since we consider binary segmentation, we use a pixel-wise *supervised* misclassification loss

$$\ell(x) = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} |x_i - y_i|, \quad \text{where the target value is } \; y_i \in \{-1, 1\}. \tag{16}$$

Results. We use the Caltech-UCSD Birds-200-2011 dataset [24] and model the magnetic field of the Ising model by a U-Net [25]. See Appendix C for details and additional results. Directly training the model for segmentation using a standard cross-entropy loss yields an average Dice score of 0.857 on the test set. In contrast, using a learned magnetic field in the Ising model, the average Dice score increases to 0.870 on the test set. Figure 2 compares two predictions from the two models. Notably, the output from the magnetization network of the Ising model exhibits clearer and sharper borders compared to the logits predicted by the plain segmentation model. This distinction is also reflected in the predictions, where the Ising model's segmentation mask appears more even and self-consistent.

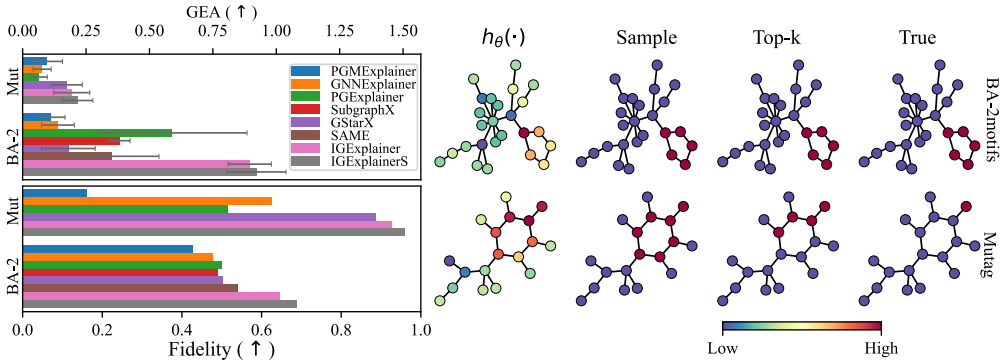

Figure 3: Left: Quantitative evaluation of IGExplainer and IGExplainerS using a synthetic dataset BA-2motifs and a real-world dataset Mutag. We report the graph explanation accuracy (GEA) and the fidelity score; higher is better for both scores. The theoretical limit of GEA is 0.43 for Mutag and 1 for BA-2motifs. **Right:** Qualitative analysis of the explanations generated by IGExplainer with the magnetic field h, the sample average, the top-k nodes, and the truth. See Appendix D.2 for baselines.

4.2 Explainability for Graph Classification

GNNs integrate information from features with the graph topology. While being a key ingredient of their success, this makes their predictions challenging to explain [26]. Nevertheless, there is a clear need for such explanations in high-stakes applications like drug design. We focus on the most studied class of problems in GNN explainability: graph classification. However, our approach, the Ising Graph Explainer (IGExplainer), is equally applicable to explain node and edge classifications.

Our Approach. Like PGExplainer [27], we derive subgraph explanations from a trained probabilistic classifier. More specifically, we consider a subgraph to be an explanation when the model's prediction on that subgraph agrees with the prediction on the whole graph. In the taxonomy of Kakkad et al. [26], this makes it a perturbation-based post-hoc method. To find such subgraph explanations, we train the external magnetic field of a ferromagnetic Ising model ($J > 0$) by minimizing the cross-entropy between predictions $y \sim p(y \mid \mathcal{G})$ for the whole graph $\mathcal{G}$ and predictions $y_\mathrm{s} \sim p(y \mid \mathcal{G}_\mathrm{s})$ for sampled subgraphs $\mathcal{G}_\mathrm{s}$. Specifically, subgraphs are sampled by subsampling nodes according to the Ising model and including all edges between those nodes. In the special case of binary classification, the cross-entropy reduces to the familiar expression

$$H(y, y_\mathrm{s}) = -\left(p \log p_\mathrm{s} + (1-p)\log(1-p_\mathrm{s})\right), \tag{17}$$

where we use the shorthand notation $p := p(y = 1 \mid \mathcal{G})$ and $p_\mathrm{s} := p(y = 1 \mid \mathcal{G}_\mathrm{s})$. To avoid the trivial solution $\mathcal{G}_\mathrm{s} = \mathcal{G}$, we normalize the external magnetic field to be zero-sum.

Results. We train our IGExplainer using a three-layer graph isomorphism network (GIN) [28] as the magnetic field $h_\theta(\cdot)$, with the cross-entropy from Equation (17) as the task-specific loss. We consider two standard explainability datasets: BA-2Motifs [27] and Mutag [29, 30]. On each dataset, we first train a classification model $\mathcal{M}$ (also a GIN network), then use $\mathcal{M}$ to compute the loss for the IGExplainer via the probability distributions p and p_s. We propose two inference procedures: IGExplainer, which uses the top-k nodes according to the trained magnetic field score, and IGExplainerS, which generates an expandability mask by performing five MCMC iterations on the Ising model.

We evaluate the explanations using two metrics: graph explanation accuracy (GEA) [30] and characteristic fidelity [31], which measures how well y_s aligns with y. In BA-2Motifs, k is set to 5, the exact number of true nodes, while in Mutag, k is set to 40% of the total graph size. Figure 3 (left) illustrates that IGExplainer and IGExplainerS both outperform all baselines in terms of GEA and fidelity, with IGExplainerS further improving accuracy through the use of Monte Carlo sampling compared to IGExplainer's static top-k selection. Figure 3 (right) visualizes the magnetic field $h_\theta(\cdot)$ for a representative graph from each dataset, where the magnetic field strength indicates node importance. We also present the average sample from IGExplainerS, highlighting its preference for sampling cohesive node groups, contributing to higher fidelity. Appendix D.4 provides details on the inference times, showing that IGExplainerS, with an average inference time of 0.02 seconds, is approximately 1000 times faster than GStarX. Additional experimental details, model architecture, and dataset descriptions are provided in Appendix D.

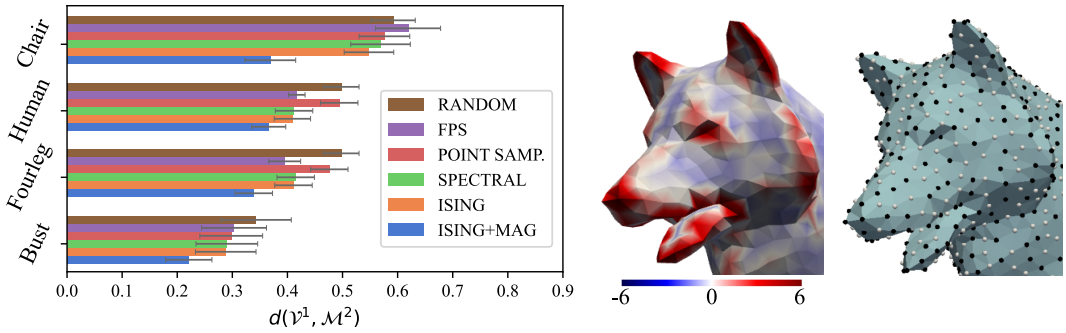

Figure 4: Left: Mean and standard deviation of the test vertex-to-mesh distance from the five-fold cross-validation of the four datasets. Here, lower is better. For an explanation of the other methods and the results as a table, see Appendix E. **Middle:** Learned magnetic field of an object. **Right:** Sampled vertices using the Ising model with a learned magnetic field. Black vertices are retained in the coarser mesh and are more important for the overall shape preservation (see the ears and nose).

4.3 3D Shape Sparsification

An object's 3D mesh tends to have a densely sampled surface, resulting in numerous redundant vertices. These redundant vertices increase the computational workload in subsequent processing stages. Consequently, mesh sparsification (representing the object with fewer vertices and edges) is a common preprocessing step, e.g., in computer graphics [32] and fluid dynamic simulations [33]. The main techniques for sparsifying a mesh are based on preserving the spectral properties of the original mesh [34, 35]. However, these methods cannot be adapted to specific downstream tasks.

We aim to create a nested mesh using a subset of nodes from the fine mesh. Unlike methods such as QSlim [36], which repositions vertices, our approach ensures continuity between mesh levels, a valuable feature for scientific applications (e.g., finite element analysis [33]). Fixing vertex positions also allows for clearer comparisons of subsampling methods by isolating the effects of sampling alone.

A triangular mesh $\mathcal{M}$ is composed of vertices $\mathcal{V}$, edges $\mathcal{E}$ and faces $\mathcal{F}$, where the faces define the triangles formed by the vertices. To sparsify the mesh, we remove selected vertices and collapse the connected edges to their nearest neighboring vertex. The distance between the vertices $\mathcal{V}^1$ of the original mesh and the surface of the coarser mesh $\mathcal{M}^2$ is defined as:

$$d(\mathcal{V}^1, \mathcal{M}^2) = \sum_{x \in \mathcal{V}^1} \min_{y \in \mathcal{M}^2} ||x - y||_2^2. \tag{18}$$

For each vertex in $\mathcal{M}^1$, the algorithm finds the nearest point in $\mathcal{M}^2$ and sums the squared distances.

Our Approach. We approach mesh sparsification as a sampling problem, using the antiferromagnetic Ising model to select vertices for the coarse mesh. This model is well-suited because its "every other" pattern helps to distribute the samples more evenly across the mesh. Additionally, the magnetic field $h_\theta(x)$ guides sampling toward regions that are most relevant to the task, i.e., shape preservation.

The 3D shapes are represented as graphs, with vertex positions as node features and relative distances as edge features, $e_{ij} = x_i - x_j$. To model the magnetic field $h_\theta(\cdot)$, we use a three-layer Euclidean GNN [37], capturing 3D geometry while remaining equivariant to rotations and translations. We also use the mesh Laplacian [38] to assess surface geometry (see Appendix E).

Results. We use the vertex-to-mesh distance in Equation (18) as our task-specific loss and train the Ising model with $\eta = 0$ in Equation (15), aiming to reduce the number of vertices in each mesh by half. As datasets, we use *bust*, *four-leg*, *human*, and *chair* [39], where each dataset contains 20 different meshes. We evaluate the model using five-fold cross-validation. Figure 4 demonstrates that the Ising model with the magnetic field (ISING+MAG) achieves the lowest point-to-mesh distance since it is trained for this specific task. ISING and Spectral coarsening (SPECTRAL) [6] perform similarly, both producing an every-other pattern, where SPECTRAL selects nodes via the graph Laplacian's largest eigenvector. While farthest point sampling [40, 41] distributes points evenly, it misses key areas. ISING+MAG combines geometric insights with effective mesh preservation. Additional details on the model, dataset, baselines, and time complexity are presented in Appendix E.

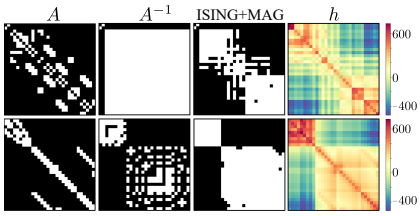

Figure 5: Sparsity patterns of A, its inverse A^{-1}, our sparse approximate inverse (50% elements, *Setting 3*), and predicted magnetic field h for two test matrices (non-zeros in white).

Table 1: Mean and standard deviation of the loss in Equation (20) over the test set for each setting and model using a single sample from the learned Ising model. Lower is better.

Model	Setting 1	Setting 2	Setting 3
ISING+MAG	**3.28** (0.12)	**2.06** (0.26)	**1.15** (1.24)
ISING	3.86 (0.12)	3.69 (0.23)	3.78 (0.42)
RANDOM	4.04 (0.16)	3.86 (0.18)	3.86 (0.30)
ONLY A	4.12 (0.08)	4.12 (0.08)	2.00 (1.89)

4.4 Sparse Approximate Matrix Inverses

Large linear systems $Az = b$, where $A \in \mathbb{R}^{n \times n}$ is a sparse matrix, are frequently solved in many scientific and technical domains. With increasing problem size, direct methods are replaced by iterative methods, e.g., Krylov subspace methods. However, if the problem is ill-conditioned, these methods require preconditioning [42] for which sparse approximate inverses (SAIs) are often suitable.

Our goal is to find a sparse approximate inverse $M \approx A^{-1} \in \mathbb{R}^{n \times n}$, acting as a right preconditioner for the sparse linear system $Az = b$. Let I be the identity matrix and $s \in \mathcal{S}$ denote the sparsity pattern of M, with $\mathcal{S}$ representing the space of all possible sparsity patterns. The pattern s specifies the m indices of non-zero entries, while $s' \in \mathbb{R}^m$ provides their corresponding values. Further, $M'(s)$ denotes the approximate inverse with sparsity pattern s and non-zeros s'. We find M by solving

$$\min_{s' \in \mathbb{R}^m} \|AM'(s) - I\|_{\mathrm{F}}, \tag{19}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. However, this approach requires an a priori assumption on s, which affects the quality of the preconditioner. Various modifications to the described sparse approximate inverse approach exist, aiming to refine the sparsity pattern iteratively [43].

Our Approach. Symmetric sparse linear systems can be represented as graphs [44] where A is viewed as the adjacency matrix of a weighted undirected graph where the matrix elements correspond to edge features. The idea is to use the Ising model to sample sparsity patterns of the SAI. However, our Ising model is defined over nodes; hence, we move the graph representation to its line graph [45]. A sample x from our model then corresponds to a proposed sparsity pattern s of the predicted SAI M. As before, we parametrize our model by θ and learn it by minimizing Equation (15) using $\eta = 0$ and

$$l(x) = \|AM(s_\theta(x)) - I\|_{\mathrm{F}}. \tag{20}$$

We aim to reduce the number of elements of an a priori sparsity pattern (possibly including all elements) by 50%. We obtain the specific SAI $M(s_\theta(x))$ by solving Equation (19) using the sampled sparsity pattern of the Ising model given its current parameters. The external magnetic field is modeled by a graph GCN [46], and we use the RLOO gradient estimator according to Equation (14). We let A and A^2 enter as edge features. Contrary to conventional approaches, our method is flexible in terms of a priori sparsity patterns. Appendices F.1 and F.2 contain further details.

Datasets. *Dataset 1* consists of 1600 synthetic binary sparse matrices of size 30×30 and *Dataset 2* consists of 1800 sparse 30×30 submatrices constructed from the SuiteSparse Matrix Collection [47], thus resembling real-world sparse matrices. See Appendix F.3 for details.

Results. We consider three experimental settings. *Setting 1* uses A and A^2 as edge features on *Dataset 1*. Here, the a priori sparsity pattern of the SAI is the same as that of A^2, and our model selects 50% of those positions for M. *Setting 2* is similar but allows for using 50% of all possible positions. *Setting 3* follows the same approach as *Setting 2* but is applied to *Dataset 2*. See Appendix F.1 for implementation details. Figure 5 shows two test samples from Setting 3, illustrating how the learned magnetic field adapts to generate an appropriate sparsity pattern even when the true inverses have different structures. More detailed results can be found in Appendix F.1. Table 1 compares the performance of our model (ISING+MAG) with three baselines: (i) an Ising model with a small constant magnetic field, tuned for the same sampling fraction as the test set (ISING), (ii) uniform random sampling from the allowed sparsity pattern with the same sampling fraction (RANDOM), and (iii) the quality of the sparse approximate inverse using only the input matrix A (ONLY A). In all cases, ISING+MAG significantly outperforms the baselines.

5 Related Work

Statistical Physics and Random Fields. The Ising model is a foundational model for magnetic systems, with generalizations such as the Potts model for multiple states [11]. The Ising model with random interactions represents spin glass systems [12], closely linked to Hopfield networks—models of associative memory that have seen renewed interest [48–50]. These have, in turn, inspired the development of Markov random fields [51]. Since we output the external magnetic field as an intermediate variable that defines an Ising model, it belongs to the class of conditional random fields [52], which are widely used in structured prediction tasks [22, 23, 53].

Combinatorial Optimization. Recent research has employed Ising models to tackle combinatorial optimization across domains such as collaborative filtering, traffic prediction, and graph learning [54–56]. These approaches infer coupling parameters and biases from historical data, which remain fixed post-inference. This method works well for static graphs, where inference minimizes energy configurations based on node values. In contrast, Salehinejad and Valaee [57] apply an Ising model to enforce sparsity and prune convolutional neural networks, treating the network as a graph and using the model as a regularizer akin to dropout. Our approach differs by introducing a parametric mapping from graph data to the magnetic field, allowing for task-specific learning of Ising parameters.

Graph Sparsification and Coarsening. Graph sparsification and coarsening aim to create a smaller graph while retaining the global structure of the original. Bravo-Hermsdorff and Gunderson [58] presents a unifying framework for both operations by reducing the graph while preserving the Laplacian pseudoinverse. Unlike our approach, theirs is algorithmic rather than learned, focusing on structural preservation without training data to adapt the reduction. Many algorithms for graph coarsening aim to preserve spectral properties [7, 59–61]. Building on top of this idea are approaches based on neural networks [8]. A graph coarsening can also be computed based on optimal transport [62], graph fusion [63], and Schur complements to obtain embeddings of relevant nodes [64]. The coarsening of graphs is also done for the scalability of GNNs [65] and dataset condensation [9, 66]. The goal is that GNNs perform similarly on the condensed data but are faster to train. Finally, Chen et al. [34] provides an overview of successful coarsening techniques for scientific computing. As for graph sparsification, spectral sparsification [67] and information-theoretic formulations [68] can be used. Graph partitioning can be done using quantum annealing by minimizing an Ising objective [69]. Batson et al. [70] provides an overview on the topic of sparsification, and Hashemi et al. [71] presents a general survey on graph reduction, including sparsification, coarsening, and condensation.

6 Discussion and Conclusion

We proposed a new method for graph subsampling by first learning the external magnetic field in the Ising model and then sampling from the resulting distribution. Our approach has shown potential in various applications and does not require the differentiability of the loss function of a given application. We achieved this using the REINFORCE Leave-One-Out gradient estimator and a carefully chosen regularization structure that allowed us to adjust the sampling fraction. Within a diverse set of experiments on four applications, we demonstrated the effectiveness of our method. In image segmentation, our approach generated segmentation masks with sharp borders. Our IGExplainer produced explanations with high accuracy and fidelity for graph classification. In mesh coarsening, the learned magnetic field increased the sampling rate in high-curvature regions while maintaining the characteristic "every other" pattern. Finally, we illustrated that our approach can learn sparsity patterns for determining sparse approximate matrix inverses via Frobenius norm minimization.

Limitations and Future Work. For our applications, the computations were fast (see Figures 19 and 20 in Appendix E), but this may change for other large-scale applications, where the sampling (incl. graph coloring) can become a bottleneck. This is something we want to address in future work. Another limitation is that the Ising model only allows for binary states, whereas some applications, e.g., image classification, require multiple classes. We believe it is possible to extend our approach to such cases by instead using the Potts model or the even more general random cluster model [72].

Acknowledgements

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Denise Pumain, editor. *Hierarchy in Natural and Social Sciences*, volume 3 of *Methodos Series*. Springer Dordrecht, 2006. 1

[2] Pierre Duhamel and Martin Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal Processing*, 19(4):259–299, 1990. 1

[3] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[4] Wolfgang Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[5] Tony Lindeberg. *Scale-space theory in computer vision*, volume 256. Springer Science & Business Media, 2013. 1

[6] David I Shuman, Mohammad Javad Faraji, and Pierre Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 64(8):2119–2134, 2015. 1, 7, 23

[7] Yu Jin, Andreas Loukas, and Joseph JaJa. Graph coarsening with preserved spectral properties. In *International Conference on Artificial Intelligence and Statistics*, pages 4452–4462. PMLR, 2020. 1, 9

[8] Chen Cai, Dingkang Wang, and Yusu Wang. Graph coarsening with neural networks. In *International Conference on Learning Representations*, 2021. 1, 9

[9] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 720–730, 2022. 1, 9

[10] Barry A Cipra. An introduction to the Ising model. *The American Mathematical Monthly*, 94 (10):937–959, 1987. 1

[11] Fa-Yueh Wu. The Potts model. *Reviews of Modern Physics*, 54(1):235, 1982. 1, 9

[12] Hidetoshi Nishimori. *Statistical physics of spin glasses and information processing: an introduction*. Clarendon Press, 2001. 1, 2, 9

[13] Jiaxin Shi, Yuhao Zhou, Jessica Hwang, Michalis Titsias, and Lester Mackey. Gradient estimation with discrete stein operators. In *Advances in Neural Information Processing Systems*, volume 35, pages 25829–25841, 2022. 1, 3

[14] Lev Davidovich Landau and Evgenii Mikhailovich Lifshitz. *Course of theoretical physics*. Elsevier, 2013. 2

[15] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021. 2, 4

[16] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017. 2

[17] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. 3

[18] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003. 3, 4

[19] Gary L Miller, Dafna Talmor, and Shang-Hua Teng. Optimal coarsening of unstructured meshes. *Journal of Algorithms*, 31(1):29–65, 1999. 3

[20] Fredrik K Gustafsson, Martin Danelljan, Radu Timofte, and Thomas B Schön. How to train your energy-based model for regression. In *Proceedings of the British Machine Vision Conference (BMVC)*, September 2020. 4

[21] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6: 721–741, 1984. 5

[22] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4): 834–848, 2017. 5, 9

[23] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015. 5, 9

[24] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Mesh segmentations. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 5

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer, 2015. 5

[26] Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. A survey on explainability of graph neural networks. *IEEE Data Engineering Bulletin*, 46(2):35–63, 2023. 6

[27] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. In *Advances in Neural Information Processing Systems*, volume 33, pages 19620–19631, 2020. 6, 18, 19

[28] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. 6, 20

[29] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, 2005. 6, 18

[30] Chirag Agarwal, Owen Queen, Himabindu Lakkaraju, and Marinka Zitnik. Evaluating explainability for graph neural networks. *Scientific Data*, 10(1):144, 2023. 6, 18

[31] Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. Graphframex: Towards systematic evaluation of explainability methods for graph neural networks. *Learning on Graphs Conference*, 2022. 6

[32] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98 (6):1031–1044, 2010. 7

[33] Jiri Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015. 7

[34] Jie Chen, Yousef Saad, and Zechen Zhang. Graph coarsening: from scientific computing to machine learning. *SeMA Journal*, pages 1–37, 2022. 7, 9

[35] Alexandros Dimitrios Keros and Kartic Subr. Generalized spectral coarsening. *arXiv preprint arXiv:2207.01146*, 2022. 7

[36] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 209–216, 1997. 7

[37] Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*, 2022. 7, 22

[38] Martin Reuter, Silvia Biasotti, Daniela Giorgi, Giuseppe Patanè, and Michela Spagnuolo. Discrete Laplace–Beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33(3):381–390, 2009. 7

[39] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3d mesh segmentation. *ACM Transactions on Graphics (TOG)*, 28(3):1–12, 2009. 7

[40] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997. 7

[41] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 7

[42] Paul Häusner, Ozan Öktem, and Jens Sjölund. Neural incomplete factorization: learning preconditioners for the conjugate gradient method. *Transactions on Machine Learning Research*, 2024. 8

[43] Marcus J Grote and Thomas Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997. 8

[44] Nicholas S Moore, Eric C Cyr, Peter Ohm, Christopher M Siefert, and Raymond S Tuminaro. Graph neural networks and applied linear algebra. *arXiv preprint arXiv:2310.14084*, 2023. 8

[45] Jens Sjölund and Maria Bånkestad. Graph-based neural acceleration for nonnegative matrix factorization. *arXiv preprint arXiv:2202.00264*, 2022. 8

[46] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020. 8, 27

[47] Timothy A. Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1–25, 2011. 8, 29

[48] Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. In *Advances in Neural Information Processing Systems*, volume 29, 2016. 9

[49] Mete Demircigil, Judith Heusel, Matthias Löwe, Sven Upgang, and Franck Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168:288–299, 2017.

[50] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, et al. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2020. 9

[51] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT Press, 2009. 9

[52] John D Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2001. 9

[53] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, 2016. 9

[54] Zhuo Liu, Yunan Yang, Zhenyu Pan, Anshujit Sharma, Amit Hasan, Caiwen Ding, Ang Li, Michael Huang, and Tong Geng. Ising-cf: A pathbreaking collaborative filtering method through efficient ising machine learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023. 9

[55] Zhenyu Pan, Anshujit Sharma, Jerry Yao-Chieh Hu, Zhuo Liu, Ang Li, Han Liu, Michael Huang, and Tony Geng. Ising-traffic: Using ising machine learning to predict traffic congestion under uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9354–9363, 2023.

[56] Chunshu Wu, Ruibing Song, Chuan Liu, Yunan Yang, Ang Li, Michael Huang, and Tong Geng. Extending power of nature from binary to real-valued graph learning in real world. In *The Twelfth International Conference on Learning Representations*, 2025. 9

[57] Hojjat Salehinejad and Shahrokh Valaee. Pruning of convolutional neural networks using ising energy model. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3935–3939. IEEE, 2021. 9

[58] Gecia Bravo-Hermsdorff and Lee Gunderson. A unifying framework for spectrum-preserving graph sparsification and coarsening. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 9

[59] Ilya Safro, Peter Sanders, and Christian Schulz. Advanced coarsening schemes for graph partitioning. *Journal of Experimental Algorithmics*, 19:1–24, 2015. 9

[60] Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*, pages 3237–3246. PMLR, 2018.

[61] Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019. 9

[62] Tengfei Ma and Jie Chen. Unsupervised learning of graph hierarchical abstractions with differentiable coarsening and optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8856–8864, 2021. 9

[63] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. In *International Conference on Learning Representations*, 2020. 9

[64] Matthew Fahrbach, Gramoz Goranci, Richard Peng, Sushant Sachdeva, and Chi Wang. Faster graph embeddings via coarsening. In *International Conference on Machine Learning*, pages 2953–2963. PMLR, 2020. 9

[65] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 675–684, 2021. 9

[66] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022. 9

[67] Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011. 9

[68] Shujian Yu, Francesco Alesiani, Wenzhe Yin, Robert Jenssen, and Jose C Principe. Principle of relevant information for graph sparsification. In *Uncertainty in Artificial Intelligence*, pages 2331–2341. PMLR, 2022. 9

[69] Hayato Ushijima-Mwesigwa, Christian FA Negre, and Susan M Mniszewski. Graph partitioning using quantum annealing on the d-wave system. In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, pages 22–29, 2017. 9

[70] Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013. 9

[71] Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B Aditya Prakash, and Wei Jin. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024. 9

[72] Cornelius Marius Fortuin and Piet W Kasteleyn. On the random-cluster model: I. introduction and relation to other models. *Physica*, 57(4):536–564, 1972. 9

[73] Adrian Kosowski and Krzysztof Manuszewski. Classical coloring of graphs. *Contemporary Mathematics*, 352:1–20, 2004. 17

[74] Aric Hagberg, Pieter J Swart, and Daniel A Schult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008. 17

[75] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 18, 20, 22, 28

[76] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 19

[77] Minh N. Vu and My T. Thai. Pgm-explainer: probabilistic graphical model explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, 2020. 19

[78] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*, pages 12241–12252. PMLR, 2021. 19

[79] Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. Gstarx: Explaining graph neural networks with structure-aware cooperative games. In *Advances in Neural Information Processing Systems*, volume 35, pages 19810–19823, 2022. 19

[80] Ziyuan Ye, Rihan Huang, Qilin Wu, and Quanying Liu. Same: Uncovering gnn black box with structure-aware shapley-based multipiece explanations. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 6442–6466. Curran Associates, Inc., 2023. 19

[81] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 19

[82] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019. 20

[83] Olga Sorkine. Laplacian mesh processing. *Eurographics (State of the Art Reports)*, 4(4), 2005. 22

[84] Rolandos Alexandros Potamias, Stylianos Ploumpis, and Stefanos Zafeiriou. Neural mesh simplification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18583–18592, 2022. 23

Appendix

The appendix first presents some details on Ising sampling and graph coloring. It then continues with extra information on our four applications. We only used a single GPU, an NVIDIA GeForce RTX 3090, for all applications and performed all experiments using less than 10 GPU hours.

Contents

A Ising Sampling

To illustrate the convergence rate of the Ising model, we plot the energy against the number of Monte Carlo iterations in Figure 6. These simulations are done on the 20 meshes in the Bust dataset of Section 4.3. We plot the mean plus one standard dedication of the model energy at each iteration. We observe that the ferromagnetic ($J = 1$) and antiferromagnetic ($J = -1$) Ising models converge fast, even though the antiferromagnetic one converges faster in just a handful of iterations.

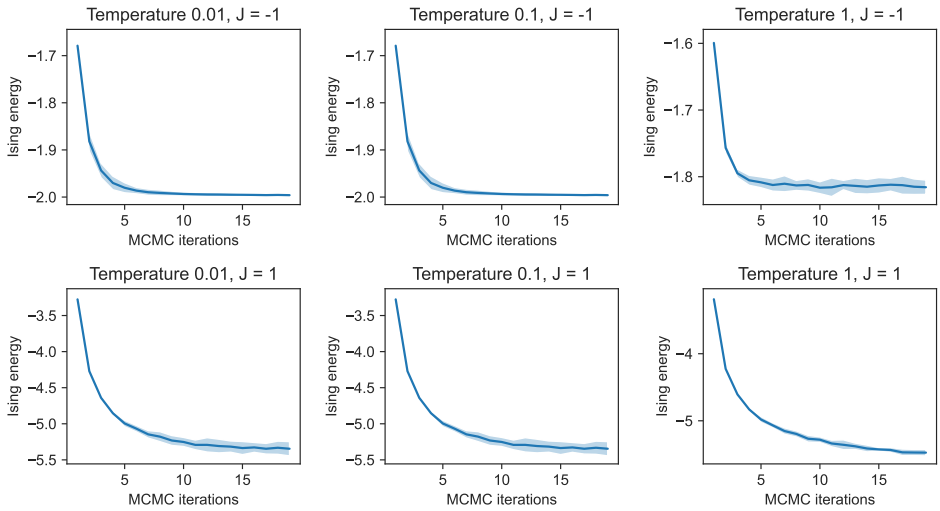

Figure 6: The mean and standard deviation of the Ising model energy against the number of Monte Carlo iterations for the ($J = 1$) or antiferromagnetic ($J = -1$) Ising model. The simulation is also done for three different temperatures: 0.01, 0.1, and 1.

B Graph Coloring

The graph coloring is based on the greedy graph coloring algorithm explained in [73]. We use the implementation of NetworkX [74]. The computational complexity of the greedy coloring method is $\mathcal{O}(m+n)$, where n is the number of vertices and m is the number of edges in the graph. Figure 7 shows the coloring time plotted against the number of vertices for the graph coarsening dataset of Section 4.3. This indicates that the time taken for coloring is insignificant compared to the time required for sampling and coarsening. However, the coloring time could potentially become problematic for exceptionally large graphs. It is worth noting that coloring an image is extremely fast and straightforward due to its simple structure.

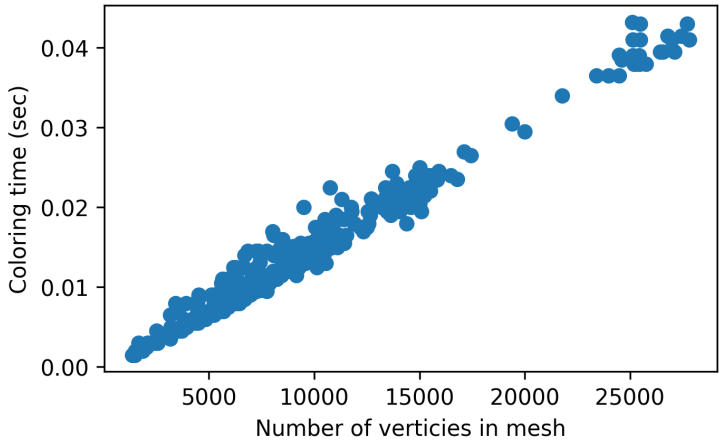

Figure 7: The time it takes to color a graph, plotted against the number of vertices in the graph.

C Binary Segmentation Details

We train a U-Net for the bird segmentation task in Section 4.1 using three down-sampling blocks, one middle layer, and three up-sampling blocks. The implementation is inspired by `https://github.com/yassouali/pytorch-segmentation/tree/master`. The dimensions of the blocks/layers are presented in Table 2.

Table 2: Dimensions of the U-Net blocks.

LAYER	INPUT DIM	OUTPUT DIM
INPUT LAYER	3	16
ENCODER 1	16	32
ENCODER 2	32	64
ENCODER 3	64	128
MID LAYER	128	128
DECODER 1	128	64
DECODER 2	64	32
DECODER 3	32	16
OUTPUT LAYER	16	1

We train the model using the Adam optimizer [75], a batch size of 32, and a learning range of $1e-4$. The dataset contains 6537 segmented images. We split the dataset in a train/val/test split using the ratios 0.8/0.1/0.1. We train the model for 300 epochs and test the model with the lowest validation error. Here, the Ising model uses a temperature of one and $J=1$. Figure 8 illustrates that the Ising segmentation model is relatively robust to noise compared to using solely a U-Net.

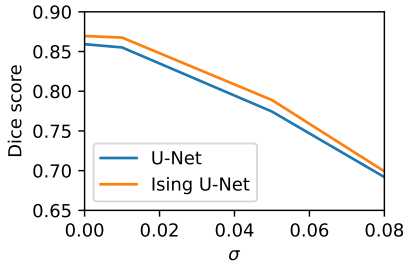

Figure 8: Dice score comparison of the two segmentation models under varying levels of noise, characterized by different standard deviations (σ), added to the input images.

D Explainability for Graph Classification Details

This section contains the details of the explaining graph classification experiments of Section 4.2.

D.1 Datasets

The *BA-2motif* dataset, first presented in [27], contains 1000 random Barabasi-Albert (BA) graphs where 50% of the graphs have a house motif attached to it, while the other half has a cycling motif attached to it. The aim is to classify which of the two a graph belongs to. All graphs in the dataset contain 25 nodes. Figure 9 depicts some samples.

The *Mutag* (or *Mutagenicity*) dataset [29] consists of 1,768 molecular graphs, categorized based on their mutagenic effects on the Gram-negative bacterium S. Although the original dataset contains 4,337 graphs, we utilize a curated version from [30], which emphasizes the presence or absence of the toxicophores: NH_2, NO_2, aliphatic halides, nitroso, and azo groups. Upon reviewing the dataset and its ground truth explanations, we observe that they are occasionally incomplete. While the labels correctly identify the relevant atoms, they often fail to account for the atomic neighborhood. For example, as illustrated in Figure 10, a chlorine (Cl) atom alone does not consistently result in

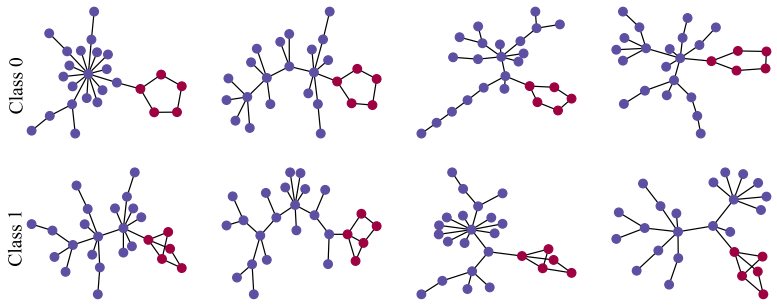

Figure 9: Samples from the two different classes in the BA-2motif dataset.

a mutagenic molecule; the mutagenicity also depends on the atomic environment around the Cl atom. However, this contextual factor is not considered in the explanation labeling of the dataset. Consequently, the absolute value of the Graph Explanation Accuracy (GEA) metric should not be overemphasized; instead, it should be interpreted in relation to other models rather than as an isolated performance indicator.

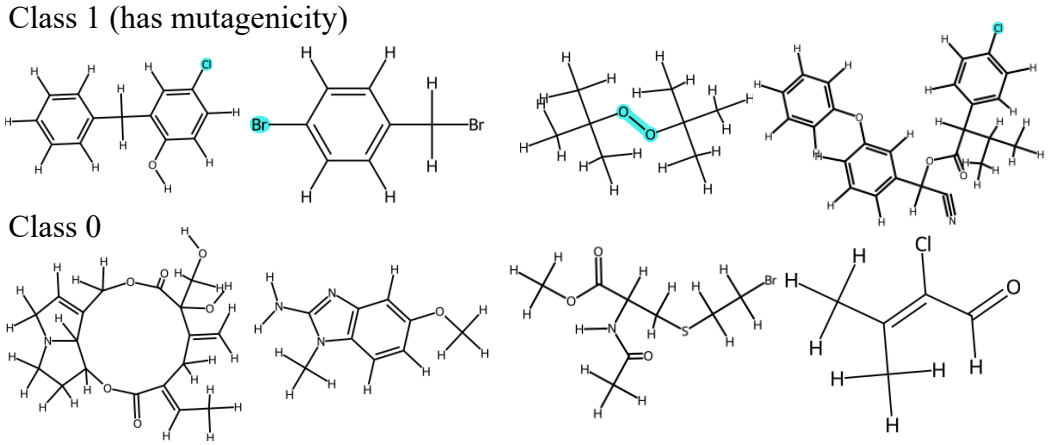

Figure 10: Samples from the two different classes in the Mutag dataset, with ground truth explanation labels highlighted for the positive (mutagenic) class. The figure shows that the provided explanations can be incomplete. For instance, chlorine (Cl) or bromine (Br) atoms attached to a carbon ring are associated with mutagenicity in the top row. However, in the bottom row, molecules containing Cl or Br are non-mutagenic, indicating that these atoms alone do not always lead to mutagenicity.

D.2 Baselines

In our experiments, we evaluate six post-hoc explainability baselines: PGExplainer [27], GNNExplainer [76], PGMExplainer [77], SubgraphX [78], GStarX [79], and SAME [80]. These methods identify significant subgraph structures influencing GNN predictions. PGExplainer uses a neural network to learn patterns across multiple data points and identify subgraphs of importance, while GNNExplainer optimizes an edge mask for each instance using gradient descent. PGMExplainer utilizes probabilistic graphical models to capture node dependencies and offer interpretable explanations of their impact on predictions. SubgraphX applies Monte Carlo Tree Search with Shapley values to systematically explore subgraphs and generate globally interpretable explanations. GStarX leverages the HN value from cooperative game theory to efficiently identify influential subgraphs by considering structure-aware insights. SAME (Structure-Aware Model Explanation) explains GNNs by analyzing the contribution of each substructure while maintaining the balance of overall model complexity. We utilize the official PyTorch Geometric [81] implementations for PGExplainer, GNNExplainer, and PGMExplainer, the implementation from [79] for GStarX and SubgraphX, and the original implementation for SAME. We follow the hyperparameters recommended by the authors.

SAME and SubgraphX do not support edge features, so they are only evaluated on the BA-2Motifs dataset.

D.3 Implementation Details

For the explainability experiments in Section 4.2, we use a three-layer GIN network [28] with a hidden dimension of 64 for the BA-2Motifs dataset. For the Mutag dataset, we use a three-layer GINE network [82], which incorporates edge features. The hidden dimension is again 64. In all experiments, the classification model and the magnetic field model share the same structure, with the only difference being that a final mean pooling layer is applied to the graph classification model.

We train the classification models using the Adam optimizer [75], with a batch size of 64 and a learning rate of 10^{-3}, for a total of 500 epochs.

For our IGExplainer model, we use again the Adam optimizer [75] with a batch size of 64 and a learning rate of 10^{-3}, and train for 300 epochs. The model with the lowest cross-entropy loss (see Equation (17)) is saved. In these explainability experiments, we evaluate the model on the training set rather than using a separate test set. The focus of explainability is to interpret and understand the behavior of the trained model, not to assess its performance on new data.

D.4 Additional Results

The time required for each model to generate an explanation is shown in Figure 11. IGExplainer and IGExplainerS are faster than all baselines except PGExplainer, which is the fastest model. IGExplainerS uses a few Monte Carlo samples to achieve the highest accuracy among the models, which makes it slightly slower than IGExplainer, which instead selects the top-k values of the magnetic field for its explanations. IGExplainerS, which provides the best performance, is still significantly faster (0.02 seconds on average for the BA-2Motif dataset) than GStarX, the slowest model, which takes an average of 25 seconds.

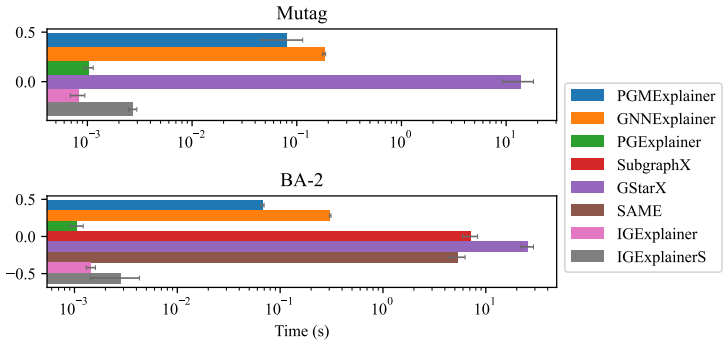

Figure 11: Inference time for all models used in the explainability experiments.

When taking samples of the IGExplainer, every sample will differ since the Ising model is stochastic. This is visualized in Figure 12 for one graph in the *BA-2motif* dataset. We also visualize several samples from the distribution and the sample average, i.e., approximately the marginal probability of including a node. Importantly, this differs from the magnetic field $h_\theta(\cdot)$.

Additional predictions from the IGExplaner on the Mutag dataset are presented in Figure 13.

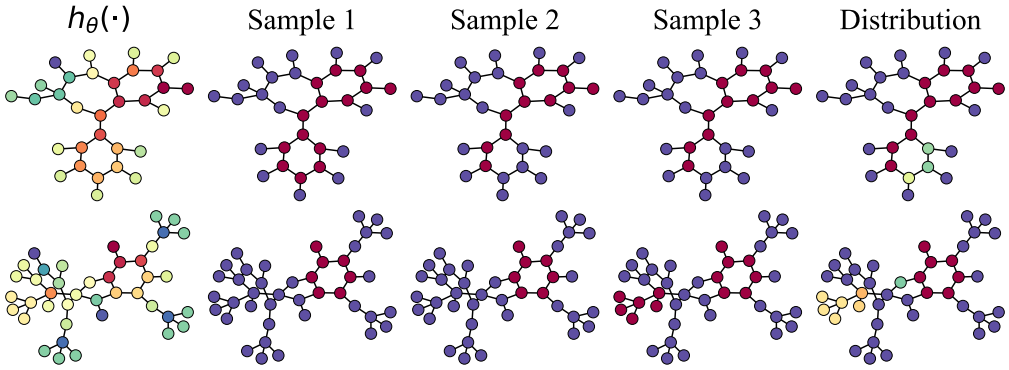

Figure 12: Samples from the trained IGExplainer. The distribution of the samples is different from the magnetic field $h_\theta(\cdot)$.

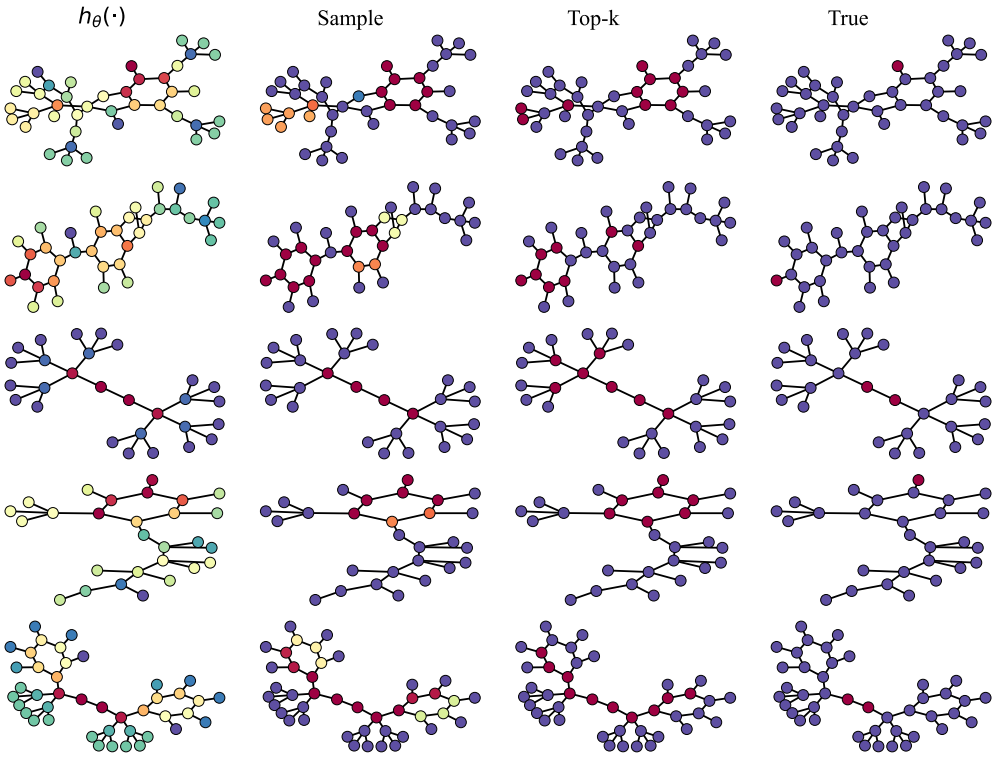

Figure 13: Output of the IGExplainer on the Mutag dataset. From left to right: the predicted magnetic field $h_\theta(\cdot)$, the sample distribution, the top-25% of nodes where $h_\theta(\cdot)$ is the score, and finally the ground truth.

E Mesh Simplification

This section contains extra information on the mesh simplification experiments of Section 4.3.

E.1 Dataset

The details of the mesh dataset are presented in Table 3.

Table 3: Details of the meshes in the dataset used in Section 4.3. The table presents the mean number and the standard deviation in parentheses.

DATASET	BUST	FOURLEG	HUMAN	CHAIR
NUMBER OF VERTICES	25662 (1230)	9778 (3671)	8036 (4149)	10154 (999)
NUMBER OF FACES	51321 (2460)	19553 (7342)	16070 (8302)	20313 (2002)

E.2 Implementation Details

The equivariant GNN is constructed using the *e3nn* library [37]. As the node input, we derive the mesh Laplacian according to

$$\mathbf{C}_{ij} = \begin{cases} w_{ij} = \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2}, & \text{if } i,j \text{ is an edge,} \\ -\sum_{j \in N(i)} w_{ij}, & \text{if } i \text{ is in the diagonal,} \\ 0, & \text{otherwise,} \end{cases} \tag{21}$$

where and α_{ij} and β_{ij} denote the two angles opposite of edge (i,j) [83]. We get a measure of the curvature of the node using

$$z_i^0 = \left\| \sum_{i \in \mathcal{N}(i)} \mathbf{C}_{ij} (\hat{r}_i - \hat{r}_j) \right\|_2^2, \tag{22}$$

where $\hat{r}_i$ is the position of node i. We use z_i^0 as the model node input. As edge attributes, we use the spherical harmonics expansion of the relative distance between node i and its neighbors j, $Y\left((\hat{r}_i - \hat{r}_j)/\|\hat{r}_i - \hat{r}_j\|\right)$ and the node-to-node distance $d_{ij} = \|\hat{r}_i - \hat{r}_j\|$. We construct an equivariant convolution by first deriving the message from the neighborhood as

$$\hat{z}_i^{k+1} = \frac{1}{n_d} \sum_{j \in \mathcal{N}(i)} z_j^k \otimes_{f_W(d_i j)} Y\left(\frac{\hat{r}_i - \hat{r}_j}{\|\hat{r}_i - \hat{r}_j\|} \right), \tag{23}$$

where n_d is the mean number of neighboring nodes of the graph. For a triangular mesh, the degree is commonly six. The operator $\otimes_{f_W(d_i j)}$ defines the tensor product with distant dependent weights $f_W(d_i j)$; this is a two-layer neural network with learnable weights w that derives the weights of the tensor product. The node features are then updated according to

$$z_i^{k+1} = z_i^k + \alpha \cdot \hat{z}_i^{k+1}, \tag{24}$$

where α is a learnable residual parameter dependent on z_i^k. We took inspiration for the model from the e3nn homepage, see https://github.com/e3nn/e3nn/blob/main/e3nn/nn/models/v2106/points_convolution.py.

We construct a two-layer equivariant network where the layer irreducible representations are [0e, 16x0e + 16x1o, 16x0e]. This means that the input (0e) consists of a vector of 16 scalars, and the hidden layers consist of vectors with 16 scalars (16x0e) together with 16 vectors (16x1o) of odd parity. This, in turn, maps to the output vector of 16 scalars (16x0e). We subtract the representation with the mean of all nodes in the graph and finally map this representation to a single scalar output. Here, we use a temperature of one and $J = -1$ for the Ising model.

For optimization, we employ the Adam optimizer [75] with a learning rate of 0.001 and a batch size of one. We train the model until convergence on the training set but with a maximum of 50 epochs. We did not use a validation set as there were no indications of overfitting.

E.3 Baselines

We compare our ISING+MAG model to five other sampling methods.

- RANDOM is regular random sampling, where every node has a 50% chance of being sampled, and the samples are independent.
- FPS stands for farthest point sampling, also sometimes called farthest point strategy. FPS selects a subset of points from a point cloud by iteratively picking the point farthest from the already chosen points, maximizing the coverage of the original set.
- POINT SAMPLER is the point sampler in [84], which uses a traditional GNN approach. It aimed to be faster than the iterative FPS and better than random sampling by using a GNN with a specific *DevConv* layer to evaluate each point's importance by analyzing its deviation from neighboring points.
- SPECTRAL is based on spectral coarsening [6], which uses the polarity of the largest Laplacian eigenvector, which, just as the Ising models, aims to find an "every other" pattern.
- ISING is the Ising model, but where the magnetic field is set to zero, resulting in a model that aims to sample in an "every other" pattern.

These methods cannot be trained for specific downstream tasks; they are inherently designed to distribute samples across the mesh evenly rather than optimize for particular applications.

E.4 Training Time

Table 4 presents the ISING+MAG and POINT SAMPLER training times. The extended training time for ISING+MAG is primarily due to the need to coarsen the mesh to compute the loss.

Table 4: Training time per epoch (in seconds) for the mesh sparsification problem. We train the ISING+MAG model for 40-60 epochs and the point sampler for 150. The largest part of the time comes from redrawing the edges in the coarse mesh.

DATASET	BUST	FOURLEG	HUMAN	CHAIR
ISING+MAG	35.22	9.32	11.82	12.99
POINT SAMPLER	0.48	0.174	0.162	0.213

E.5 Extra Results

The outcome of the experiments presented as a bar chart in Figure 4 is presented in Table 5.

Table 5: Mean and standard deviation of the test vertex-to-mesh distance from the five-fold cross-validation of the four datasets. Lower is better.

DATASET	BUST	FOURLEG	HUMAN	CHAIR
ISING+MAG	**0.221** (0.042)	**0.339** (0.034)	**0.366** (0.031)	**0.369** (0.046)
ISING	0.288 (0.055)	0.411 (0.034)	0.409 (0.033)	0.548 (0.045)
RANDOM	0.343 (0.064)	0.498 (0.032)	0.498 (0.032)	0.592 (0.040)
SPECTRAL	0.290 (0.056)	0.415 (0.034)	0.412 (0.034)	0.569 (0.054)
POINT SAMPLER	0.298 (0.057)	0.476 (0.034)	0.494 (0.034)	0.576 (0.046)
FPS	0.303 (0.059)	0.395 (0.029)	0.417 (0.015)	0.619 (0.059)

The results presented in Table 5 show that the Ising model with the magnetic field (ISING+MAG) surpasses all other models regarding the point-to-mesh distance. This outcome aligns with expectations, given that our model is trained for this specific task. Their results are virtually identical when comparing the Ising model without the magnetic field (ISING) to Spectral coarsening (SPECTRAL). This is anticipated since both approaches result in an every-other pattern. In SPECTRAL, nodes are

chosen based on the polarity of the largest eigenvector of the graph Laplacian, which divides the nodes into two equally large subgroups.

While the farthest point sampling (FPS) method ensures that samples are distributed across the graph, it has its limitations. In some datasets, it may outperform ISING and SPECTRAL. However, it is important to note that FPS does not guarantee an "every other" pattern and may inadvertently miss nodes in critical areas of the mesh.

Overall, the Ising model with the magnetic field (ISING+MAG) outperforms all other models. Similar to ISING and SPECTRAL, it samples an "every other" pattern. However, leveraging a geometry-aware network also learns to identify nodes crucial for maintaining mesh shape. This behavior, which defaults to using the "every other" pattern and selectively samples nodes based on the geometric shape of surrounding nodes, contributes to the model's effectiveness.

Examples of how the Ising model learns important areas of the mesh are presented in Figures 14, 15, 16, and 17. Here, we visualize the magnetic field from the test split from the four datasets. An illustration of the course meshes after sampling is presented in Figure 18.

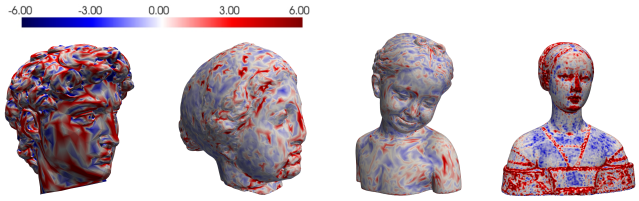

Figure 14: Examples of the magnetic field from the bust dataset (test set).

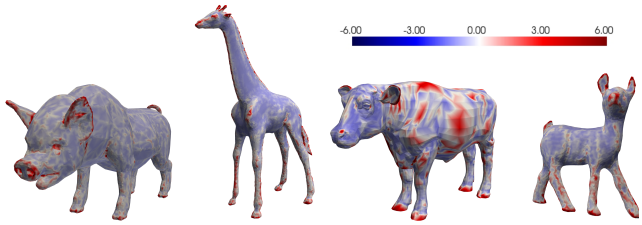

Figure 15: Examples of the magnetic field from the fourleg dataset (test set).

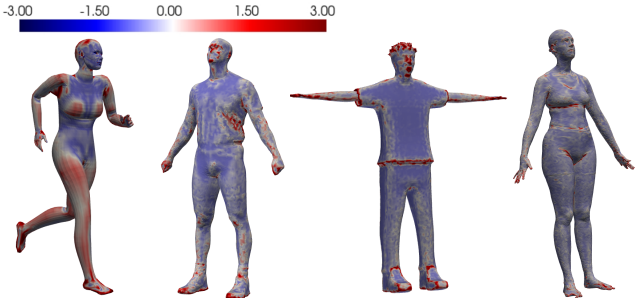

Figure 16: Examples of the magnetic field from the human dataset (test set).

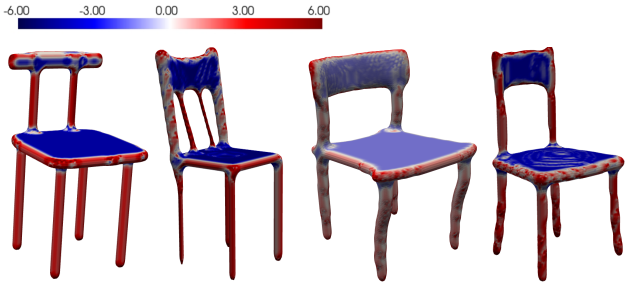

Figure 17: Examples of the magnetic field from the chair dataset (test set).

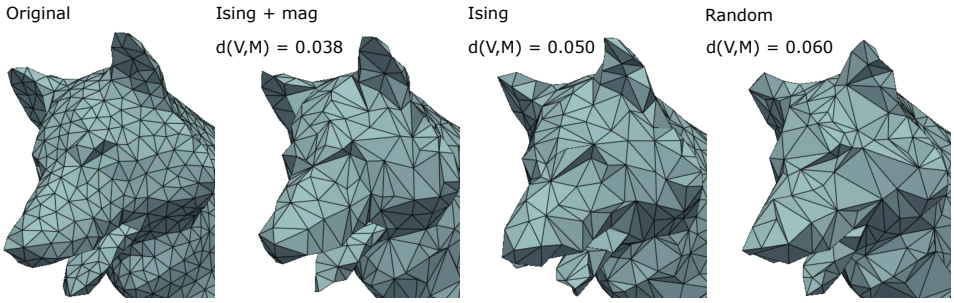

Figure 18: The coarsened mesh using random subsampling, Ising subsampling, and learned Ising subsampling of the nodes. The point-to-mesh distance is reduced from 0.06 to 0.038 when going from the random subsampling to the learned Ising graph. The time it takes to sample and create the course mesh is random, 1.37 sec, Ising 0.67 sec, and Ising+mag 1.31 sec. Random sampling takes longer due to a more complicated course mesh creation.

Complexity. Given the large graphs in mesh processing, evaluating time complexity is crucial. We measure the time complexity for coarsening and sampling to complement the results in Table 5. Figure 19 shows the sampling time alone, excluding mesh reconstruction, as a function of the mesh size. Notably, only random sampling and the neural network-based point sampler are faster than the Ising model. The relatively efficient scaling of the Ising model is due to two factors: (i) the parallelism enabled by the Metropolis-Hastings algorithm through graph coloring, and (ii) the fast convergence of the Ising model, which requires only a few iterations of the Metropolis-Hastings algorithm.

Figure 20 illustrates the combined time for sampling and coarsening. It shows that the Ising model's coarsening time is lower than that of random sampling. This is because coarsening is less efficient with unevenly distributed samples. Additionally, coarsening is the major time factor, while the sampling time is negligible.

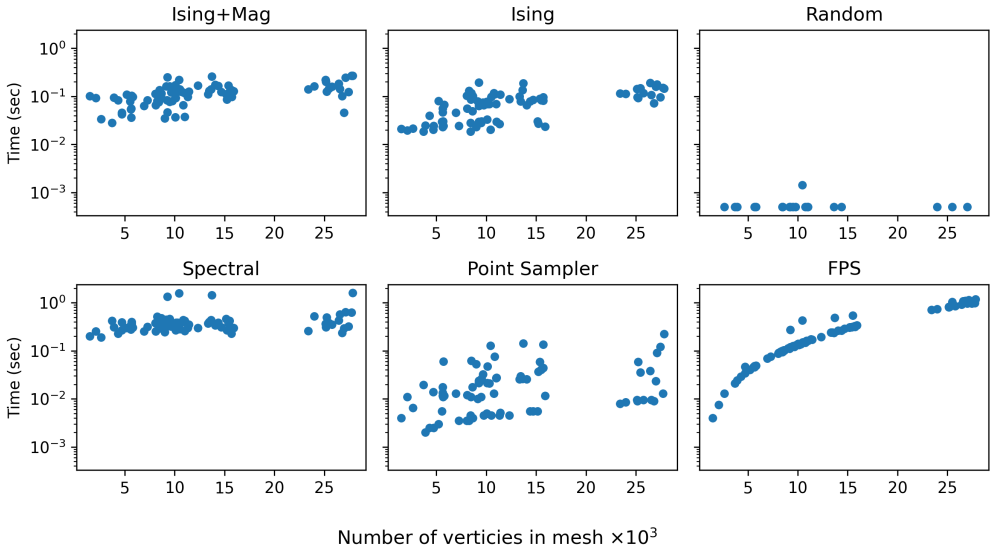

Figure 19: The time it takes to get sample nodes to include in the coarse graph.

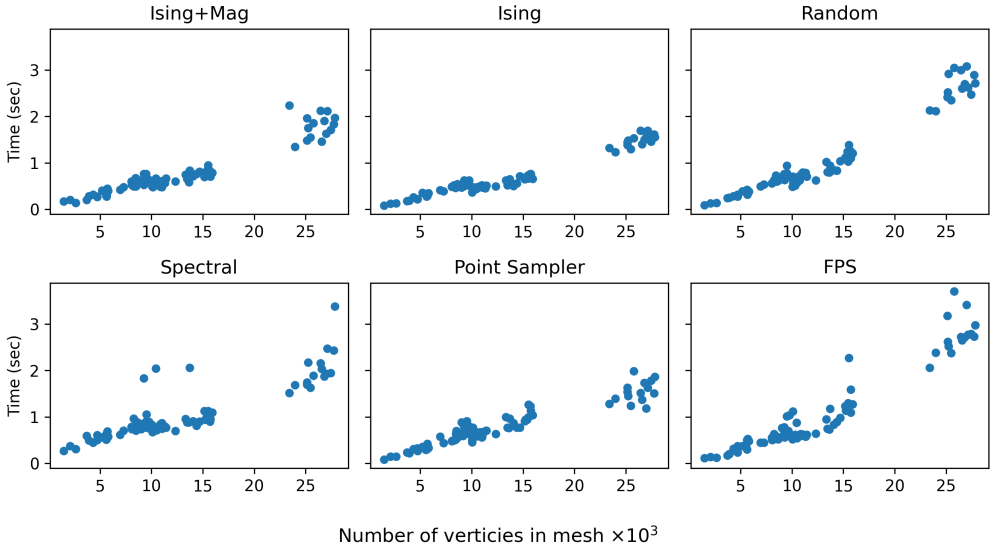

Figure 20: The time it takes to sample and coarsen the graph, plotted against the number of vertices in the graph.

F Sparse Approximate Matrix Inverses

F.1 Results and Implementation Details

With our initial model architecture, the graph representation induces an a priori assumption on the sparsity pattern of the predicted sparse approximate inverse in which we aim to select 50% of the elements in the sparsity pattern of $A + A^2$. Our approach is, however, not limited to the choice of an a priori sparsity pattern. Implementation-wise, we lift this constraint by adding an extra edge feature consisting of an all-ones $n \times n$ matrix in the graph representation of the input matrix, allowing for a selection of 50% of all matrix elements in the sparse approximation of the inverse. As the a priori sparsity pattern is already incorporated into the graph representation of the input matrix, this renders our method flexible in terms of choice of a priori sparsity pattern, which in our case should be denser than the desired sparsity level. This flexibility is expected to be especially important for computational efficiency in use cases involving large matrix dimensions.

Given the current state of the trainable parameters, two samples from the Ising model are required during training to obtain the gradient estimate. Evaluation is carried out using a single sample. In both cases, the sample x from the Ising model produced after T MCMC iterations is passed to the final layer, which solves the the n optimization problems,

$$s_k^* = \underset{s_k' \in \mathcal{R}^{m_k}}{\arg\min} \|AM_k(s_k) - I_k\|_2^2, \tag{25}$$

computing the values $s_k^* \in \mathbb{R}^m$ of the non-zero entries of each column M_k of M. Thus, the final output is the predicted sparse approximate inverse M. Empirically, $T = 3$ is found to be sufficient.

Independent on which dataset is used, 60% of the samples are used for training, and the remaining 40% are divided equally between a validation set and a test set. We use the magnetic field-dependent regularization described in Section 3 to optimize for a sampling fraction of 50% of the a priori sparsity pattern. In all settings, the sampling fraction converges to an average value close to 50% with the proposed regularization scheme. The mean recorded sampling fraction on the test dataset in *Setting 1* and *Setting 2* is 52.4% and 50.9%, respectively. The corresponding sampling fraction in *Setting 3* is 49.8%. The baseline methods are tuned to allow for the same sampling fraction during evaluation.

Results for two samples from the test dataset in *Setting 3* are visualized in Figure 5. The first sample (top) shows the predicted sparsity pattern when the true inverse is dense. For this sample, the recorded loss is 2.61 (ISING+MAG) compared to 3.57 (Ising), 4.18 (Random) and 4.12 (Only A). As discussed in Figure 1, the magnitude of the local magnetic field influences the sampling probability in that region, which can be seen by comparing the output magnetic field to the obtained sparsity pattern for both samples. A zero magnetic field in the Ising model can be interpreted as a prior on the sparsity pattern in which 50% of the elements in each row and column of the matrix are selected, resulting in an inverse approximation in which the nonzero elements are evenly distributed across the given matrix. For the second sample (bottom), the sparsity optimized for is lower than the sparsity of the true inverse. We choose to display this sample as the results clearly illustrate that the learned magnetic field indeed results in an Ising model from which the sampled sparsity pattern successfully avoids the most unimportant regions in the true inverse for the particular input matrix. For this sample, the recorded loss is 1.02 (ISING+MAG) compared to 3.75 (Ising), 3.76 (Random) and 4.06 (Only A).

The current baselines include comparisons to uniform random sampling and an Ising model without the learned magnetic field key to our approach, both restricted to the allowed sampling fraction. This restriction is key to fair comparisons since including more nonzeros can improve the performance metric. With this in mind, comparison to assuming a sparsity pattern corresponding to the sparsity pattern of input matrix A is a good sanity check, but in this case worse performance can be expected if the sparsity of the predicted approximate inverse is lower.

F.2 Graph Convolutional Neural Network

Here, we use the simple and deep graph convolutional networks from Chen et al. [46] where we set the strength of the initial residual connection α to 0.1 and θ, the hyperparameter to compute the strength of the identity, to 0.5. The network contains four layers, each with a hidden dimension of 64. We utilize weight sharing across these layers. The dimension of the output of the final convolutional layer is 64. This output is subtracted with the mean of the node values before a final linear layer reduces the size to one.

We employ the Adam optimizer [75] with a learning rate of 0.01 for optimization. We train the model until convergence on the training set but with a maximum number of 300 epochs, as there were no indications of overfitting. Here, the Ising model uses a temperature of one and $J = -0.4$.

The training times per epoch were 1 minute for *Setting 1* and three minutes for *Setting 2* and *Setting 3*, respectively. These models converged in less than 50 epochs.

F.3 Datasets

Dataset 1. This is a synthetic dataset containing 1600 binary matrices. Each matrix is real, symmetric, and of dimensionality 30×30. The matrices in the dataset are automatically generated based on the principle that the probability of a nonzero element increases towards the diagonal, with 100% probability of nonzero diagonal elements. The upper right and lower left corners have a very low probability of being nonzero, and in between these elements and the diagonal, the probability of a nonzero element varies according to a nonlinear function. The mean sparsity is 83% zero-elements in the generated dataset. The maximum sparsity allowed in the data-generating process is 96%, and the determinant of each matrix is bounded between 0.001 and 50. Figure 21 shows sparsity patterns of four matrices in the final dataset.

The dataset is available upon request and will be released together with the code used for preprocessing and generation.

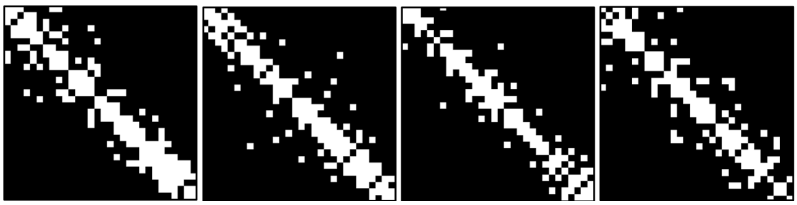

Figure 21: Example of samples from *Dataset 1*. Nonzero elements are represented in white.

Dataset 2. This is a synthetic dataset containing 1800 submatrices of size 30×30 constructed from the SuiteSparse Matrix Collection [47]. Since the original matrices in the dataset are often denser closer to the diagonal, the submatrices are constructed by iterating a window of size 30×30 along the diagonals, such that no submatrix overlaps another submatrix. To ensure that each submatrix is symmetric, only the upper triangular part of the matrix is used to create each matrix. Each submatrix is scaled such that the absolute value of the maximum element is one, and submatrices with all values below 10^{-6} are removed from the dataset. The mean sparsity in the final dataset is 89%. The maximum sparsity allowed in the data-generating process is 96%, and the determinant of each matrix is bounded between 0.001 and 50. Figure 22 shows sparsity patterns of four matrices in the final dataset.

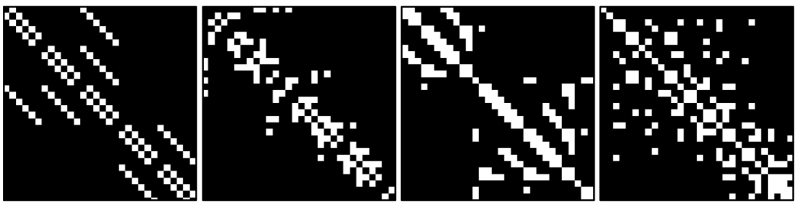

Figure 22: Example of sparsity patterns of samples from *Dataset 2*. Nonzero elements are represented in white.

The original dataset is under the CC-BY 4.0 License. Our modified dataset is available upon request and will be released with the code used for preprocessing and generation.