
Jointly Modelling Uncertainty and Diversity for Active Molecular Property Prediction

Anonymous Author(s)

Anonymous Affiliation

Anonymous Email

Abstract

1
2 Molecular property prediction is a fundamental task in AI-driven drug discovery.
3 Deep learning has achieved great success in this task, but relies heavily on abundant
4 annotated data. However, annotating molecules is particularly costly because it
5 often requires lab experiments conducted by experts. Active Learning (AL) tackles
6 this issue by querying (*i.e.*, selecting) the most valuable samples to annotate, ac-
7 cording to two criteria: *uncertainty* of the model and *diversity* of data. Combining
8 both criteria (a.k.a. hybrid AL) generally leads to better performance than using
9 only one single criterion. However, existing best hybrid methods rely on some
10 trade-off hyperparameters for balancing uncertainty and diversity, and hence need
11 to carefully tune the hyperparameters in each experiment setting, causing great
12 annotation and time inefficiency. In this paper, we propose a novel AL method
13 that jointly models uncertainty and diversity without the trade-off hyperparameters.
14 Specifically, we model the joint distribution of the labeled data and the model
15 prediction. Based on this distribution, we introduce a Minimum Maximum Prob-
16 ability Querying (MMPQ) strategy, in which a single selection score naturally
17 captures how the model is uncertain about its prediction, and how dissimilar the
18 sample is to the currently labeled data. To model the joint distribution, we adapt
19 the energy-based models to the non-Euclidean molecular graph data, by learning
20 chemically-meaningful embedding vectors as the proxy of the graphs. We perform
21 extensive experiments on binary classification datasets. Results show that our
22 method achieves superior AL performance, outperforming existing methods by a
23 large margin. We also conduct ablation studies to verify different design choices.

24 1 Introduction

25 AI-driven drug discovery is an important application of machine learning. In drug discovery pipeline,
26 a fundamental step is to use computational methods to predict the molecular properties (*e.g.*, toxicity
27 and binding specificity) of candidate compounds [1, 2]. Recently, deep learning models have achieved
28 great success in molecular property prediction [1, 3–5], but their high performance relies on a large
29 amount of annotation. However, annotating molecules is particularly time-consuming and costly,
30 since it often requires lab experiments or complex computation [3, 6].

31 One promising way to alleviate this problem is Active Learning (AL) [7], which aims at finding a
32 strategy for iteratively querying (*i.e.*, selecting) the most valuable data samples to annotate, so as to
33 maximize model performance under a low annotation budget. AL strategies query samples mainly
34 based on two criteria: *uncertainty* of the model [8], and *diversity* of queried data [9]. Strategies taking
35 into account both criteria (a.k.a. hybrid strategies) are recently shown to outperform methods based on
36 only uncertainty or diversity in many learning tasks [10–12]. Existing best hybrid methods generally
37 rely on some *trade-off hyperparameters* for balancing uncertainty and diversity [11–15]. For example,
38 WAAL [12] requires manually-tuned coefficients to obtain a weighted sum of its uncertainty and
39 diversity terms. EADA [13] relies on two selection ratios for its two-step selection process. These
40 trade-off hyperparameters are crucial to the AL performance and hence need to be carefully tuned for
41 each experiment setting.

42 However, tuning trade-off hyperparameters can cause substantial inefficiency in AL. For one thing,
43 since these hyperparameters have a large influence on the outcome of corresponding AL strategies,
44 the selected samples under different choices of the hyperparameters often vary a lot, and thus the
45 total annotation cost needed for tuning will greatly exceed the budget. For another, the tuning
46 process can take a long time, since each AL experiment iterates between query selection and
47 model (re)training for several rounds. **One possible way of addressing this drawback is to tune
48 the hyperparameters in advance on a dataset with small annotation cost. However, this would be
49 problematic when the dataset used for tuning is not representative. Indeed, for molecular property
50 prediction, finding a representative dataset for tuning would be difficult, since the properties of
51 interest vary a lot.**

52 In this paper, we propose a novel AL strategy that naturally takes into account uncertainty and
53 diversity without the need of trade-off hyperparameters. Our strategy is based on a joint distribution
54 $q(x, y) \triangleq p(y|x)p(x)$, which contains information about both uncertainty and diversity: $p(y|x)$ is the
55 prediction distribution of the model (with input x and prediction y), which is widely used to define
56 uncertainty metrics [7, 8, 16]; $p(x)$ is the density of the currently annotated data, which is shown to
57 be useful for identifying samples that can effectively increase data diversity [11, 12, 17].

58 Specifically, our strategy operates by first maximizing $q(x, y)$ via varying y , and then minimizing
59 $\max_y q(x, y)$ via varying x . We thus name our strategy Minimum Maximum Probability Querying
60 (MMPQ). Importantly, we show that the selection score of MMPQ can be viewed as the product of
61 two terms — the first term leads to samples on which the model has low prediction confidence, while
62 the second favors samples that are dissimilar to labeled data. In this way, the selected samples are
63 naturally those that the model is most uncertain about, while at the same time being able to increase
64 the data diversity.

65 For modelling the joint distribution, we propose to use an Energy-Based Model (EBM) [18, 19], since
66 it can explicitly output the desired probability value. For training the EBM, we need to tackle one key
67 challenge in our setting: the variable x in the joint distribution has a non-Euclidean data structure (*i.e.*,
68 a molecule graph), which renders the commonly-used EBM training scheme inapplicable [19, 20].
69 To address this challenge, we take a learned embedding vector z as a proxy of the non-Euclidean
70 input x , which allows us to train the EBM on z and y with the commonly-used EBM training
71 scheme. Specifically, inspired from [21, 22], we learn the embeddings by training an autoencoder
72 to reconstruct the input SMILES strings, which is an expert-defined sequence representation of
73 molecules. The EBM is trained by Denoising Score Matching (DSM) [20, 23], *i.e.*, to learn the “Stein
74 score” [24] of $q(x, y)$, which has been shown to be an efficient and robust EBM training scheme.

75 To evaluate our MMPQ strategy, we apply it to actively train a commonly-used Graph Neural
76 Network (GNN) [4] on various benchmark datasets of **binary molecular property classification**.
77 Extensive results show that MMPQ enables the GNN to achieve high performance with limited
78 annotation budget, significantly outperforming other competitive AL methods. We conduct ablation
79 studies to verify different design choices of our method. In particular, we show that the uncertainty
80 and diversity terms make complementary contributions to the good performance: the diversity term
81 is important in early iterations of AL, while the uncertainty term is essential in later iterations.
82 Anonymized code is available at <https://anonymous.4open.science/r/MMPQ-5E8D/>.

83 2 Related Works

84 **Molecular property prediction** is a critical step in drug discovery [1, 2, 5]. Traditional methods (*e.g.*,
85 based on density function theory [25]) are too slow to be applied in practice. To resolve this problem,
86 deep learning methods [1, 3, 4, 26, 27] have been widely proposed, which can be categorized into
87 two types: (1) descriptor-based methods [26, 27] that represent the input molecules as expert-crafted
88 molecular descriptors (*e.g.*, fingerprints [28]), and (2) GNN-based methods [1, 3, 4] that directly take
89 molecule graphs as input. As found in [1, 5], GNNs generally outperform descriptor-based methods,
90 and thus this work focuses on GNN-based molecular property prediction.

91 **Active learning** improves annotation efficiency by iteratively querying samples based on two criteria:
92 uncertainty of the task learner [8, 16, 29, 30], and/or diversity of queried data [9, 17, 21]. Uncertainty-
93 based methods define various uncertainty metrics for querying data [8, 10, 16, 30], while diversity-
94 based methods aim to find a representative subset of the whole dataset by querying diverse samples [9,
95 21]. Compared to using only uncertainty or diversity, recent works find that combining the two

96 criteria (a.k.a. hybrid methods) leads to better performance [10, 11, 13–15]. However, existing hybrid
 97 approaches generally need to balance uncertainty and diversity via some trade-off hyperparameters.
 98 For example, WAAL [12] uses manually-tuned coefficients to weight its uncertainty term and
 99 diversity term. EADA [13] adopts a two-stage querying approach, where each stage requires a
 100 prefixed selection ratio. We note that EADA also trains an EBM for active selection. Our method
 101 differs from EADA mainly in 3 aspects. First, the motivation of EADA to adopt EBMs is to
 102 identify out-of-distribution samples, while we use EBMs for modelling the distribution capturing
 103 both uncertainty and diversity. Second, they train their EBM via contrastive divergence [31], while
 104 we use denoising score matching [20]. Third, they need two separate selection steps with different
 105 selection scores, and require two hyperparameters to trade off between uncertainty and diversity,
 106 while we only have one single selection score that naturally captures both uncertainty and diversity.

107 Apart from the above, some other hybrid methods rely on trade-off hyperparameters during their
 108 model training process [11, 14, 15]. We note that there is an existing hybrid strategy, BADGE [10],
 109 that is also free from trade-off hyperparameters like ours. However, BADGE [10] assumes that task
 110 learner’s prediction is a faithful proxy of the ground-truth label. This may not hold in early AL
 111 iterations on (typically) small molecule datasets, since the task learner would be inaccurate due to
 112 limited training data.

113 **Energy-based models** are a class of powerful methods of explicit generative modeling. Recently,
 114 some works [32–35] leverage EBMs for modelling molecular data. To tackle difficulties caused by
 115 the discrete nature of molecule graphs, [32] leverages a dequantization technique, and [33] designs a
 116 diffusion process based on stochastic differential equations. Different from [32, 33], we propose to
 117 train our EBM on a continuous embedding space of molecules. On the other hand, [34, 35] focuses on
 118 molecule conformation generation, which is essentially a continuous problem, since the conformation
 119 of a molecule is represented by the 3D space coordinates of its atoms.

120 3 Preliminaries

121 **Problem Setting.** We consider batch-mode pool-based active learning [7], a practical setting for
 122 deep models. In each AL round, a batch of samples from the unlabeled pool \mathcal{D}_U are queried according
 123 to a strategy, annotated by an oracle, and added to the labeled pool \mathcal{D}_L . The updated \mathcal{D}_L is then used
 124 to train the task learner. A more formal description of this setting is in Appx. A.1.

125 **Notations.** A molecule is represented as a graph $G = (V, E)$, with nodes V and edges E corre-
 126 sponding to atoms and chemical bonds. As in [4, 36–38], we are interested in n binary molecular
 127 properties (e.g., toxicity), which are denoted by a label vector $\mathbf{y} = (y_1, \dots, y_n) \in \{0, 1\}^n$, where
 128 $y_i = 1$ or 0 means the molecule has the i -th property or not. A task learner $h(\cdot)$ is trained to predict
 129 the properties. The i -th output of the task learner, $h(G)_i$, specifies a distribution $p(y_i|G)$ over the
 130 predicted label of the i -th property of G , which is essentially a Bernoulli distribution with success
 131 probability $h(G)_i$ (denoted as $\text{Ber}(h(G)_i)$).

132 **Energy-Based Models.** EBMs [18] specify probability density or mass functions as follows:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}, \quad (1)$$

133 where $\mathbf{x} \in \mathbb{R}^D$ is a random sample, $E_\theta(\mathbf{x})$ is the *energy function* with learnable parameters θ ,
 134 and $Z_\theta = \int \exp(-E_\theta(\mathbf{x})) d\mathbf{x}$ is a normalizing constant. By learning θ , we can use an EBM to
 135 approximate a real data distribution, i.e., $p_\theta \approx p_{\text{data}}$.

136 **Denosing Score Matching.** DSM [23, 39] is an efficient approach for training EBMs. Here, the
 137 “(Stein) score” of a distribution $f(\mathbf{x})$ is defined as the log-probability’s first-order gradient function
 138 w.r.t. \mathbf{x} , i.e., $\nabla_{\mathbf{x}} \log f(\mathbf{x})$. DSM first disturbs $p_{\text{data}}(\mathbf{x})$ with a pre-defined noise distribution $p_N(\tilde{\mathbf{x}}|\mathbf{x})$,
 139 and then trains the EBM via

$$\mathbb{E}_{\substack{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}) \\ \tilde{\mathbf{x}} \sim p_N(\tilde{\mathbf{x}}|\mathbf{x})}} \left[\frac{1}{2} \|\nabla_{\mathbf{x}} \log p_\theta(\tilde{\mathbf{x}}) - \nabla_{\mathbf{x}} \log p_N(\tilde{\mathbf{x}}|\mathbf{x})\|_2^2 \right]. \quad (2)$$

140 With a proper $p_N(\tilde{\mathbf{x}}|\mathbf{x})$, we can easily obtain $\nabla_{\mathbf{x}} p_N(\tilde{\mathbf{x}}|\mathbf{x})$ [39].

141 4 Method

142 4.1 The Minimum-Maximum-Probability Query Strategy

143 Our proposed query strategy is based on the *joint distribution* of two key probability distributions
 144 used in existing works. The first is the prediction distribution of the task learner, *i.e.*, $p(y = \hat{y}|G)$
 145 (abbreviated as $p(\hat{y}|G)$), which is widely used to define different uncertainty metrics [7, 8, 16].
 146 The second is the distribution of currently labeled pool \mathcal{D}_L , denoted as $p_L(G)$. As shown in
 147 [11, 12, 17, 21], $p_L(G)$ is useful for identifying samples that are dissimilar to the labeled ones, and
 148 annotating these samples effectively increases data diversity. Inspired by these works, we propose to
 149 model the joint distribution of $p(y = \hat{y}|G)$ and $p_L(G)$.

150 Formally, let $q(G, \hat{\mathbf{y}})$ denote the joint distribution:

$$q(G, \hat{\mathbf{y}}) \triangleq p(\hat{\mathbf{y}}|G)p_L(G), \quad (3)$$

151 where we use the boldface $\hat{\mathbf{y}}$ because we may be interested in more than 1 tasks. Note that $\hat{\mathbf{y}}$ is a
 152 random variable following $p(\hat{\mathbf{y}}|G)$, not the ground-truth label of G .

153 Then, we perform active selection by first maximizing $q(G, \hat{\mathbf{y}})$ via varying $\hat{\mathbf{y}}$ for each single G , and
 154 then selecting a batch of G that minimizes the obtained $\max_{\hat{\mathbf{y}}} q(G, \hat{\mathbf{y}})$. Denote the selected batch as
 155 $\mathcal{B} = \{G_1, \dots, G_b\}$. Our strategy is formalized as:

$$\mathcal{B} = \arg \min_{G_1, \dots, G_b \in \mathcal{D}_U} \left(\max_{\hat{\mathbf{y}}} q(G_1, \hat{\mathbf{y}}), \dots, \max_{\hat{\mathbf{y}}} q(G_b, \hat{\mathbf{y}}) \right). \quad (4)$$

156 We name our strategy Minimum-Maximum-Probability Querying (MMPQ). The whole active learning
 157 process with MMPQ strategy is summarized in Appx. A.4.

158 4.1.1 MMPQ as a Tuning-Free Hybrid Strategy

159 Here we show that MMPQ naturally captures both uncertainty of the task learner and diversity w.r.t.
 160 the whole data space in a tuning-free manner. First, from Eqn. (4) and Eqn. (3), we can see that the
 161 selection score of MMPQ is:

$$\max_{\hat{\mathbf{y}}} q(G, \hat{\mathbf{y}}) = \left(\max_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}|G) \right) p_L(G). \quad (5)$$

162 Then, let $p^M = \max_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}|G)$, and it can be seen that the MMPQ strategy essentially selects data
 163 with smaller p^M and smaller $p_L(G)$.

164 ► **Uncertainty.** Smaller p^M corresponds to samples that the task learner is less confident about.
 165 Specifically, let $\hat{\mathbf{y}}^* = (\hat{y}_1^*, \dots, \hat{y}_n^*)$ denote the prediction that achieves p^M (*i.e.*, $p^M = p(\hat{\mathbf{y}}^*|G)$),
 166 and $\hat{\mathbf{y}}' = (\hat{y}'_1, \dots, \hat{y}'_n)$ denote any other prediction that is different from $\hat{\mathbf{y}}^*$. Note that, since
 167 $p(\hat{\mathbf{y}}^*|G) + \sum_{\hat{\mathbf{y}}' \in \{0,1\}^n, \hat{\mathbf{y}}' \neq \hat{\mathbf{y}}^*} p(\hat{\mathbf{y}}'|G) = 1$, smaller p^M means that p^M is closer to the second-largest
 168 (and all other) predictions, implying that the task learner is more uncertain about its prediction on
 169 molecule G .

170 ► **Diversity.** Smaller $p_L(G)$ means that G lies in low-density regions of the distribution of currently
 171 labeled data, and hence is dissimilar to the labeled data. Thus, querying those with small $p_L(G)$
 172 increases the diversity of the obtained labeled pool \mathcal{D}_L^t [11, 12, 17, 21].

173 Based on above reasoning, samples with lowest selection score (*i.e.*, those taken by the arg min
 174 operation in Eqn. (4)) are naturally those the model is most uncertain about, while at the same time
 175 being able to increase data diversity. As such, MMPQ does not need a hyperparameter to trade off
 176 between uncertainty and diversity.

177 4.1.2 Implementation of MMPQ

178 Since MMPQ is based on the value of $q(G, \hat{\mathbf{y}})$, we need to model $q(G, \hat{\mathbf{y}})$ with an *explicit* deep
 179 generative model. In particular, we instantiate an Energy-Based Model (EBM) using a neural network,
 180 since EBMs have been shown to be quite expressive and stable in distribution modelling [39, 40].
 181 Formally, $q(G, \hat{\mathbf{y}})$ is modelled by

$$q(G, \hat{\mathbf{y}}) = \frac{\exp(-E(G, \hat{\mathbf{y}}))}{Z}, \quad (6)$$

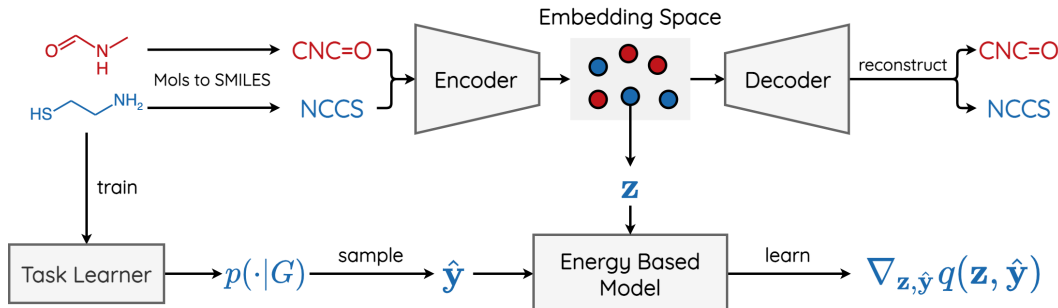


Figure 1: Model design and data flow. Colored in blue are inputs and outputs of the labeled pool and corresponding objective. Those corresponding to unlabeled pool are in red.

182 where $E(G, \hat{y})$ is the energy value given by the EBM, and Z is a normalizing constant.

183 In this subsection, we focus on how to implement MMPQ with the EBM, and thus here we assume
 184 that the EBM is already trained and fixed. Model design and training of the EBM will be presented
 185 later in Sec. 4.2.

186 From Eqn. (6), we have

$$\begin{aligned} & \arg \min_G \left(\max_{\hat{y}} q(G, \hat{y}) \right) \\ &= \arg \min_G \left(\max_{\hat{y}} (\log q(G, \hat{y}) + \log Z) \right) = \arg \min_G \left(-\min_{\hat{y}} E(G, \hat{y}) \right) = \arg \max_G \left(\min_{\hat{y}} E(G, \hat{y}) \right). \end{aligned} \quad (7)$$

187 This reveals that we can implement MMPQ based on the learned energy values, without the need to
 188 calculate the normalizing constant Z .

189 One may argue that, $\min_{\hat{y}} E(G, \hat{y})$ would be difficult to compute for large n , as it involves all
 190 2^n possible combinations of $(\hat{y}_1, \dots, \hat{y}_n)$. We show in Appx. A.2 that, based on the conditional
 191 independence assumption of labels [41], $\min_{\hat{y}} E(G, \hat{y})$ can be computed in a task-wise manner.

192 4.2 Model Design and Training of the EBM

193 4.2.1 Model Design

194 Designing an EBM for learning $q(G, \hat{y})$ is not trivial, since the two variables have different data
 195 structure: G is an attributed graph, while \hat{y} is a vector. Moreover, learning EBMs for attributed
 196 graphs is itself a challenging open problem, due to the non-Euclidean and discrete nature [32, 42].

197 To address the above issues, we propose to embed molecules graphs G into a learned embedding space,
 198 and then build the EBM model on \hat{y} and embeddings \mathbf{z} (see Fig. 1). Inspired by Sinha *et al.* [21], we
 199 learn the space by training an Auto-Encoder (AE) to reconstruct its inputs. However, due to graph
 200 isomorphism, directly reconstructing molecule graphs is difficult [22, 43, 44]. We thus propose to
 201 train the AE to reconstruct the molecules' SMILES strings [45], as shown in Fig. 1. SMILES is a
 202 [sequence representation](#) of molecules, where the sub-strings correspond to chemically-meaningful
 203 substructures in molecules (*e.g.*, functional groups). Such a sequence-based reconstruction task
 204 enables the auto-encoder to learn molecules embeddings without struggling to reconstruct graphs.

205 Formally, let $\text{Enc}(\cdot)$ and $\text{Dec}(\cdot)$ denote the encoder and decoder respectively, and let $\text{Sml}(\cdot)$ denote
 206 the operation of retrieving the SMILES string of a molecule (which can be easily pre-computed
 207 using open-sourced cheminformatics libraries). Then, for a molecule G , the ground-truth and the
 208 reconstructed SMILES strings are

$$S \triangleq \text{Sml}(G), \quad \hat{S} \triangleq \text{Dec}(\text{Enc}(\text{Sml}(G))). \quad (8)$$

209 For learning high-quality embeddings, we use both labeled and unlabeled data to train the AE:

$$L_{\text{rec}} = \mathbb{E}_{G \in \mathcal{D}_L} [d(S, \hat{S})] + \mathbb{E}_{G' \in \mathcal{D}_U} [d(S', \hat{S}')], \quad (9)$$

210 where $d(\cdot, \cdot)$ is a distance between sequence pairs.

211 In the rest of this paper, we take $\mathbf{z} \triangleq \text{Enc}(\text{Sml}(G))$ as a proxy of G in some cases, and use \mathbf{x} to
 212 denote the tuple $(\mathbf{z}, \hat{\mathbf{y}})$, which is implemented by concatenating \mathbf{z} and $\hat{\mathbf{y}}$.

213 Following previous works [39, 40], we instantiate the EBM as a ‘‘score net’’ $s_\theta(\mathbf{x})$, which learns the
 214 score of the target distribution $q(\mathbf{x})$, *i.e.*, $\nabla_{\mathbf{x}} \log q(\mathbf{x})$. When $s_\theta(\mathbf{x})$ is trained, we use summation to
 215 approximate integral over $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ (see Appx. A.3). An alternative choice is to approximate the
 216 energy function $E_\theta(\mathbf{x})$, which however is more difficult than modeling the score, as experimentally
 217 shown in Sec. 5.3.3.

218 4.2.2 Model Training

219 We train the EBM $s_\theta(\mathbf{x})$ via denoising score matching. With the noise $p_N(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2, I)$,
 220 we have $\nabla_{\mathbf{x}} p_N(\tilde{\mathbf{x}}|\mathbf{x}) = -\frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}$ [39]. Then, the DSM objective is:

$$L_{\text{DSM}} = \mathbb{E}_{\substack{\mathbf{x} \in \mathcal{D}_L, \\ \tilde{\mathbf{x}} \sim \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2)}} \left[\left\| s_\theta(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2} \right\|_2^2 \right], \quad (10)$$

221 where we slightly abuse the notation, using $\mathbf{x} \in \mathcal{D}_L$ to denote $\mathbf{x} \in \{(\text{Enc}(\text{Sml}(G)), \hat{\mathbf{y}}) | G \in \mathcal{D}_L\}$.

222 Note that the second term in the target distribution (Eqn. (3)) is the density of the labeled data only.
 223 Therefore, in Eqn. (10), we calculate L_{DSM} only on the labeled pool \mathcal{D}_L (*cf.* the reconstruction
 224 objective in Eqn. (9)).

225 One challenge of calculating L_{DSM} is that it requires $(G, \hat{\mathbf{y}})$ pairs i.i.d. sampled from $q(G, \hat{\mathbf{y}})$, but
 226 we do not have such samples at hand. To address this challenge, we propose a two-step sampling
 227 method: first randomly pick G from \mathcal{D}_L ; then draw a sample $\hat{\mathbf{y}} = \{\hat{y}_1, \dots, \hat{y}_n\}$ from $p(\hat{\mathbf{y}}|G)$, which
 228 can be implemented by drawing a $\hat{y}_i \sim \text{Ber}(h(G)_i)$ for all i (under the conditional-independence
 229 assumption of labels).

230 The EBM and the AE are jointly trained via

$$L_{\text{joint}} = L_{\text{DSM}} + L_{\text{rec}}. \quad (11)$$

231 Pseudo code of model training process is summarize in Appx. A.4.

232 5 Experiments

233 5.1 Experiment Setup

234 We run experiments under the batch-mode pool-based AL setting (elaborated in Appx. A.1). The
 235 labeled pool is initialized by randomly selecting 10% samples of the entire training; the initial
 236 unlabeled pool is the rest 90%. Then 10 AL rounds are performed; in each round, an unlabeled
 237 batch of 4% samples of the entire training set is queried, so the total annotation budget is 50%
 238 of the training set. We use the BACE, BBBP, HIV and SIDER datasets from the widely-used
 239 MoleculeNet benchmark [5] (also included in the Open Graph Benchmark [46]). Statistics and
 240 detailed descriptions of the datasets are in Appx. A.5. Following [4], we use *scaffold split*, with
 241 train/val/test = 80%/10%/10%. We use AUROC as the performance metric, as suggested in [5].
 242 Please refer to Appx. A.6 for implementation details.

243 5.2 Active Learning Performance

244 We compare MMPQ against following 8 baselines, with **U**, **D** and **H** denoting Uncertainty-based,
 245 Diversity-based and Hybrid methods respectively: **Random** (random selection), **Entropy (U)** (se-
 246 lecting samples with the largest prediction entropy), **MC-Dropout (U)** [8], **CoreSet (D)** [9], **ASGN**
 247 **(D)** [47], **BADGE (H)** [10], **WAAL (H)** [12], **EADA (H)** [13]. Among them, ASGN is the only
 248 existing method that investigates AL in molecular property prediction. BADGE, WAAL and EADA
 249 are representative hybrid methods, and are the state-of-the-arts on many image classification datasets.
 250 We note that there are other hybrid methods [11, 14, 15], but their code is not or only partly released.
 251 We fail to reproduce their results, and hence do not include them for comparison.

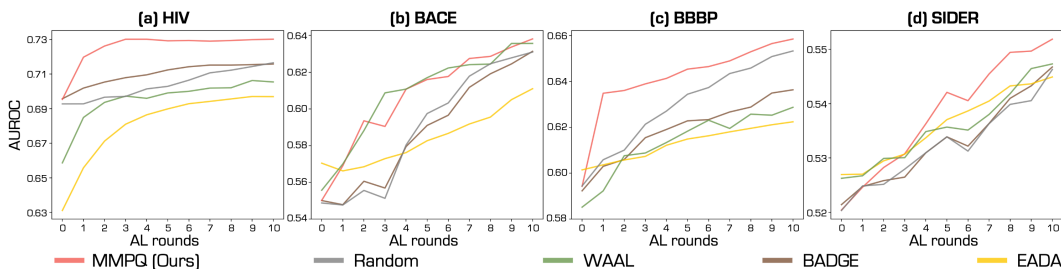


Figure 2: Active learning performance of MMPQ (ours) and baseline hybrid methods. “Round 0” corresponds to the performance on initial labeled pool. Note y-axis has an off-set for clear comparison.

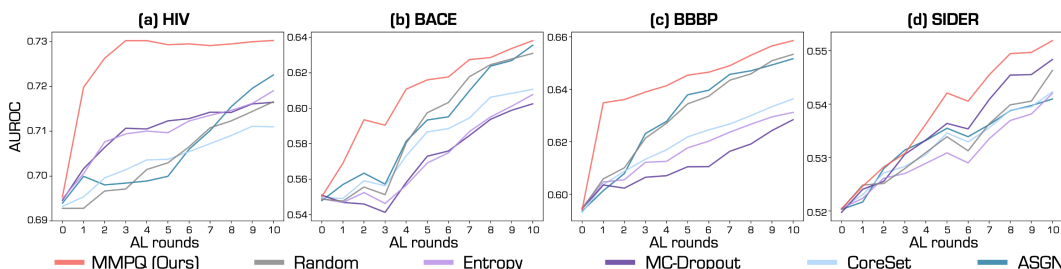


Figure 3: Active learning performance of MMPQ (ours) and uncertainty-based or diversity-based methods. “Round 0” corresponds to the performance on initial labeled pool. Note y-axis has an off-set for clear comparison.

252 To avoid cluttered presentation, we show results of baseline hybrid methods in Fig. 2, and those of
 253 uncertainty-based or diversity-based methods in Fig. 3. In both figures, we include results of our
 254 MMPQ and the Random baseline.

255 **Results:** From the figures we can see that our MMPQ outperforms the baselines on all 4 datasets.
 256 Specifically, on HIV, our MMPQ achieves 0.7302 AUROC in the 3-rd active round (using only 22%
 257 annotations of the entire training set), which is very close to the performance of using 100% of the
 258 annotations (0.7344). This also explains why the performance of MMPQ almost saturates after the
 259 3-rd round. Furthermore, the performance of hybrid methods requiring trade-off hyperparameters, *i.e.*,
 260 WAAL and EADA, is not stable. In particular, though WAAL achieves performance comparable to
 261 our proposed MMPQ on BACE, it performs quite unsatisfactorily on HIV. Similarly, EADA performs
 262 well on SIDER but is the worst baseline on HIV. By contrast, our method achieves consistently
 263 superior performance. One may note that the performance of WAAL and EADA at round 0 is quite
 264 different from that of other methods. The reason is that, in other methods, the task learner is only
 265 trained with classification loss on the currently labeled data. Contrarily, in WAAL and EADA, apart
 266 from classification loss, the task learner is also trained with some auxiliary loss (*i.e.*, adversarial loss
 267 in WAAL, and free-energy alignment loss in EADA). Therefore, even though the training data at
 268 round 0 are the same across all methods, the WAAL and EADA can have different performance.

269 5.3 Ablation Studies

270 Here we conduct ablative experiments on HIV, which is the *largest* dataset used (see Tab. 1).

271 5.3.1 Uncertainty or Diversity Only

272 In Sec. 4.1.1, we show that our MMPQ strategy captures both uncertainty and diversity through the
 273 two terms in Eqn. (5) respectively. Here we ablatively study the effectiveness of the two terms. Since
 274 we have only 1 target property on HIV, we use \hat{y} instead of $\hat{\mathbf{y}}$. Specifically, under the setup described
 275 in Sec. 5.1, we compare our MMPQ with another two strategies based on the trained EBM:

- 276 • The **U.O.** strategy that considers **Uncertainty Only**: querying data with minimum $p^M =$
 277 $\max_{\hat{y}} p(\hat{y}|G)$. Let \hat{y}^* denote the predicted label that achieves p^M . Then, based on the learned

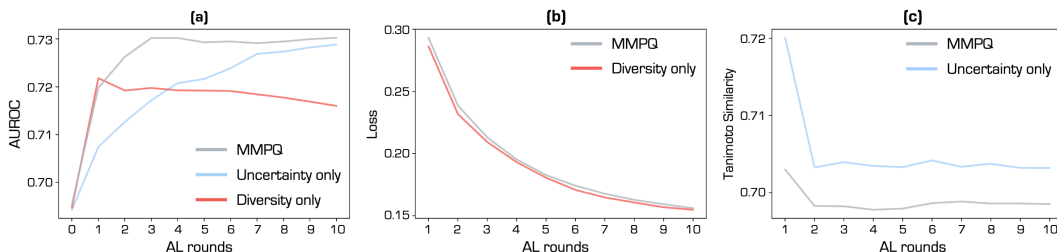


Figure 4: (a) AL performance of MMPQ, U.O. and D.O. strategies. (b) Mean ground-truth-label loss of data queried by MMPQ or D.O. strategy. (c) Mean average Tanimoto similarity of data queried by MMPQ or U.O. strategy.

278 EBMs, p^M can be calculated by:

$$p^M = \frac{\exp(-E(G, \hat{y}^*))}{\sum_{\hat{y} \in \{0,1\}} \exp(-E(G, \hat{y}))}. \quad (12)$$

279 • The **D.O.** strategy that considers **Diversity Only**: querying those with minimum $p_L(G)$:

$$p_L(G) \propto \sum_{\hat{y} \in \{0,1\}} \exp(-E(G, \hat{y})). \quad (13)$$

280 In Fig. 4 (a), as AL proceeds, the performance of the U.O. strategy rises slower than that of MMPQ
 281 or the D.O. strategy, though the final performance of U.O. (at round 10) is as good as MMPQ.
 282 On the other hand, the D.O. strategy reaches a peak very quickly (*i.e.*, at the 2-nd AL round),
 283 but then its performance degrades as more data are annotated for training. One possible reason
 284 for such degradation is that the data queried in later rounds cannot provide the task learner with
 285 useful information about the learning task. Adding these data to the training pool may lead to
 286 overfitting, since more data means more training iterations. This shows that uncertainty and diversity
 287 are complementary to each other — diversity is important in early AL stages, while uncertainty is
 288 critical in later stages. Interestingly, this corroborates the finding in [48].

289 Furthermore, we then dig deeper into the advantages of the MMPQ strategy, by examining how
 290 the two terms in Eqn. (5) affect the queried data. For studying the effectiveness of the uncertainty-
 291 based term $\max_{\hat{y}} p(\hat{y}|G)$, we examine whether the queried data of the MMPQ strategy have higher
 292 uncertainty than those of the D.O. strategy (since MMPQ and D.O. only differ in this term). Note
 293 that, since HIV has only 1 property of interest, this term becomes $p(\hat{y}|G)$. For measuring uncertainty,
 294 distribution-based metrics such as entropy and classification margin are often used. However, for
 295 binary classification (as our case), these metrics are equivalent to selecting those with smallest
 296 $p(\hat{y}|G)$. Therefore, instead of these metrics, we adopt the ground-truth-label loss, as used in [30].
 297 For the t -th AL round, we calculate the mean loss of data in the current queries \mathcal{D}_L^t . As shown in
 298 Fig. 4 (b), compared with the D.O. strategy, queries of our MMPQ have larger ground-truth-label
 299 loss, suggesting larger uncertainty of the task learner.

300 For the diversity-based term $p_L(G)$, we examine whether queries of MMPQ have smaller chemical
 301 similarity (*i.e.*, larger diversity) than those of U.O. strategy (since of MMPQ and U.O. only differ in
 302 this term). We adopt the Tanimoto similarity [49], which is a widely used expert-defined molecular
 303 similarity metric. Formally, let T_{ij} denote the Tanimoto similarity between molecule G_i and G_j , we
 304 calculate the mean Average Similarity (mAS) among molecules in \mathcal{D}_L^t :

$$mAS = \frac{1}{N_L^t(N_L^t - 1)} \sum_{G_i, G_j \in \mathcal{D}_L^t, j \neq i} T_{ij}. \quad (14)$$

305 Fig. 4 (c) shows that queries of MMPQ are less chemically similar than those of U.O., implying
 306 larger diversity.

307 5.3.2 Robustness of Energy Calculation

308 In this part, we investigate the robustness of MMPQ w.r.t. how energy is calculated. Specifically, we
 309 consider the choice of zero-energy point, and the number of points for approximating the integral
 310 (*i.e.*, K in Eqn. (18)).

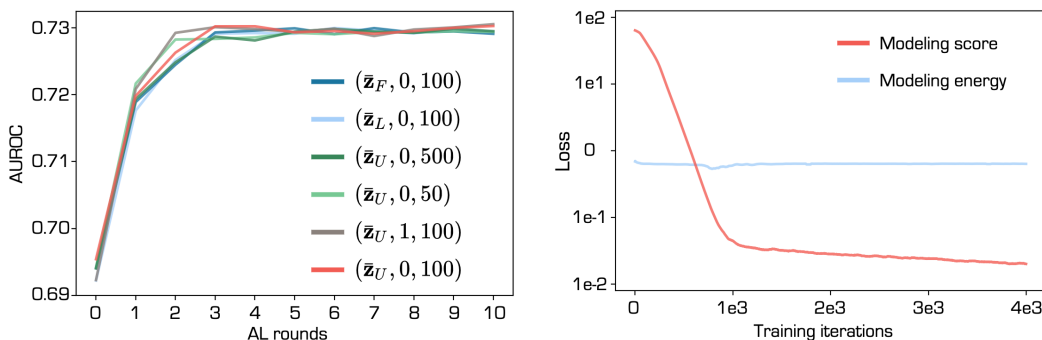


Figure 5: Performance of different $(\mathbf{z}_0, \hat{y}_0, K)$. **Figure 6:** DSM loss of modeling score or energy.

311 In above experiments, we set the zero-energy point as $(\mathbf{z}_0 = \bar{\mathbf{z}}_U, \hat{y}_0 = 0)$, where $\bar{\mathbf{z}}_U$ is the where $\bar{\mathbf{z}}_U$
 312 is the mean embedding of the unlabeled pool, and let $K = 100$. We name this setting the “default
 313 setting”, and denote it with the triple $(\bar{\mathbf{z}}_U, 0, 100)$.

314 Then, based on $(\bar{\mathbf{z}}_U, 0, 100)$, we vary one of the three hyperparameters while keeping the other two
 315 unchanged, and run AL experiments under the setup described in Sec. 5.1. Specifically, we set (1)
 316 $\mathbf{z}_0 \in \{\bar{\mathbf{z}}_L, \bar{\mathbf{z}}_F\}$, where $\bar{\mathbf{z}}_L$ and $\bar{\mathbf{z}}_F$ are the mean embedding of the Labeled pool and that of the Full
 317 training set; (2) $\hat{y}_0 = 1$; (3) $K \in \{50, 500\}$. Fig. 5 shows the AL performance of the above settings
 318 and the default one. We can see that the AL performance of different settings are similar, which
 319 demonstrates that the MMPQ strategy is robust to the above hyperparameters.

320 5.3.3 Implementing EBM by Modeling Energy

321

322 As introduced in Sec. 4.2, we instantiate the EBM as a score net learning the score of the true
 323 distribution. An alternative is to implement the EBM as an energy net modelling the energy function.

324 We try this alternative in our experiments, but find that the training process fails to converge. Specifi-
 325 cally, the energy net is also built on the embedding space of the AE, and has the same architecture as
 326 the score net, except that the final layer has a output dimension of 1 and a ReLU activation (since the
 327 energy is a non-negative scalar). The training objective of the energy net (denoted as E_θ) is

$$\mathbb{E}_{\substack{\mathbf{x} \in \mathcal{D}_L, \\ \tilde{\mathbf{x}} \sim \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma)}} \left[\left\| -\nabla_{\tilde{\mathbf{x}}} E_\theta(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2} \right\|_2^2 \right]. \quad (15)$$

328 Fig. 6 shows the loss curve (in a log scale) under best tuned hyperparameters (*i.e.*, those yielding
 329 lowest loss). For comparison, the loss curve on HIV of our score net used in the MMPQ experiment
 330 in Sec. 5.2 is also given. We can see that, even with the best tuned hyperparameters, the training
 331 process of E_θ cannot converge well.

332 6 Conclusion and Limitation

333 We propose Maximum Minimum Probability Querying (MMPQ), a hybrid active learning method
 334 for molecular property prediction, without the need of manually trading off between uncertainty and
 335 diversity. The strategy is based on an EBM that models the joint distribution of labeled data and
 336 task learner’s prediction. The EBM is built in an embedding space learned by an auto-encoder that
 337 reconstructs molecules’ SMILES string. We propose training the EBM via denoising score matching.
 338 Once the EBM is trained, MMPQ selects data according to one single selection criterion that naturally
 339 captures uncertainty of the task learner and high diversity w.r.t. in the data space.

340 One limitation of our approach is that it is only applicable to binary classification tasks. This is
 341 because we use the maximum prediction probability as an uncertainty metric, which is equivalent
 342 to the commonly-used entropy metric only in the binary case. When extending to categorical
 343 classification or regression, our used metric may not well reflect model uncertainty. In Appx. A.8, we
 344 discuss one possible solution to address this limitation by utilizing a new uncertainty metric.

References

- [1] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019. 1, 2
- [2] Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Edward W Lowe. Computational methods in drug discovery. *Pharmacological reviews*, 66(1):334–395, 2014. 1, 2
- [3] 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. 1, 2
- [4] W Hu, B Liu, J Gomes, M Zitnik, P Liang, V Pande, and J Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. 2, 3, 6, 15
- [5] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018. 1, 2, 6, 14, 15
- [6] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24):5441–5451, 2018. 1
- [7] Burr Settles. Active learning literature survey. 2009. 1, 2, 3, 4, 13
- [8] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017. 1, 2, 4, 6
- [9] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. 1, 2, 6
- [10] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2019. 1, 2, 3, 6
- [11] Kwanyoung Kim, Dongwon Park, Kwang In Kim, and Se Young Chun. Task-aware variational adversarial active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8166–8175, 2021. 1, 2, 3, 4, 6
- [12] Changjian Shui, Fan Zhou, Christian Gagné, and Boyu Wang. Deep active learning: Unified and principled method for query and training. In *International Conference on Artificial Intelligence and Statistics*, pages 1308–1318. PMLR, 2020. 1, 2, 3, 4, 6
- [13] Binhui Xie, Longhui Yuan, Shuang Li, Chi Harold Liu, Xinjing Cheng, and Guoren Wang. Active learning for domain adaptation: An energy-based approach. *AAAI conference on artificial intelligence*, 2022. 1, 3, 6
- [14] Beichen Zhang, Liang Li, Shijie Yang, Shuhui Wang, Zheng-Jun Zha, and Qingming Huang. State-relabeling adversarial active learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8753–8762. IEEE Computer Society, 2020. 3, 6
- [15] Shuo Wang, Yuexiang Li, Kai Ma, Ruhui Ma, Haibing Guan, and Yefeng Zheng. Dual adversarial network for deep active learning. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pages 680–696. Springer, 2020. 1, 3, 6
- [16] Jongwon Choi, Kwang Moo Yi, Jihoon Kim, Jinho Choo, Byoungjip Kim, Jinyeop Chang, Youngjune Gwon, and Hyung Jin Chang. Vab-al: incorporating class imbalance and difficulty with variational bayes for active learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021. 2, 4
- [17] Daniel Gissin and Shai Shalev-Shwartz. Discriminative active learning. *arXiv preprint arXiv:1907.06347*, 2019. 2, 4
- [18] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006. 2, 3

- 398 [19] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint*
399 *arXiv:2101.03288*, 2021. 2
- 400 [20] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score
401 matching. *Journal of Machine Learning Research*, 6(4), 2005. 2, 3
- 402 [21] Samrath Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning.
403 In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5971–5980.
404 IEEE Computer Society, 2019. 2, 4, 5
- 405 [22] Orion Dollar, Nisarg Joshi, David AC Beck, and Jim Pfandtner. Attention-based generative
406 models for de novo molecular design. *Chemical Science*, 2021. 2, 5, 15
- 407 [23] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural*
408 *computation*, 23(7):1661–1674, 2011. 2, 3
- 409 [24] Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit
410 tests. In *International conference on machine learning*, pages 276–284. PMLR, 2016. 2
- 411 [25] Pierre Hohenberg and Walter Kohn. Inhomogeneous electron gas. *Physical review*, 136(3B):
412 B864, 1964. 2
- 413 [26] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural
414 networks. In *International Conference on Learning Representations*, 2016. 2
- 415 [27] Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. Smiles-bert:
416 Large scale unsupervised pre-training for molecular property prediction. In *Proceedings of*
417 *the 10th ACM International Conference on Bioinformatics, Computational Biology and Health*
418 *Informatics*, 2019. 2
- 419 [28] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical*
420 *information and modeling*, 50(5):742–754, 2010. 2
- 421 [29] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of
422 ensembles for active learning in image classification. In *Proceedings of the IEEE conference on*
423 *computer vision and pattern recognition*, pages 9368–9377, 2018. 2
- 424 [30] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proceedings of the*
425 *IEEE/CVF conference on computer vision and pattern recognition*, pages 93–102, 2019. 2, 8
- 426 [31] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likeli-
427 hood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages
428 1064–1071, 2008. 3
- 429 [32] Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. Graphebm: Molecular graph genera-
430 tion with energy-based models. *arXiv preprint arXiv:2102.00546*, 2021. 3, 5
- 431 [33] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs
432 via the system of stochastic differential equations. In *Proceedings of the 39th International*
433 *Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*,
434 pages 10362–10383. PMLR, 17–23 Jul 2022. 3
- 435 [34] Shitong Luo, Chence Shi, Minkai Xu, and Jian Tang. Predicting molecular conformation via
436 dynamic graph score matching. *Advances in Neural Information Processing Systems*, 34, 2021.
437 3
- 438 [35] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular
439 conformation generation. In *International Conference on Machine Learning*, pages 9558–9568.
440 PMLR, 2021. 3
- 441 [36] Yaqing Wang, Abulikemu Abuduweili, Quanming Yao, and Dejing Dou. Property-aware
442 relation networks for few-shot molecular property prediction. *Advances in Neural Information*
443 *Processing Systems*, 34, 2021. 3, 15
- 444 [37] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. Motif-based graph
445 self-supervised learning for molecular property prediction. *Advances in Neural Information*
446 *Processing Systems*, 34, 2021. 15
- 447 [38] Shengchao Liu, Meng Qu, Zuobai Zhang, Huiyu Cai, and Jian Tang. Structured multi-task
448 learning for molecular property prediction. In *International Conference on Artificial Intelligence*
449 *and Statistics*, pages 8906–8920. PMLR, 2022. 3

- 450 [39] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data
451 distribution. *Advances in Neural Information Processing Systems*, 32, 2019. 3, 4, 6
- 452 [40] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable
453 approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages
454 574–584. PMLR, 2020. 4, 6
- 455 [41] Haoran Wang, Weitang Liu, Alex Bocchieri, and Yixuan Li. Can multi-label classification
456 networks know what they don’t know? *Advances in Neural Information Processing Systems*,
457 34, 2021. 5
- 458 [42] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon.
459 Permutation invariant graph generation via score-based generative modeling. In *International
460 Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020. 5
- 461 [43] Jaechang Lim, Seongok Ryu, Jin Woo Kim, and Woo Youn Kim. Molecular generative
462 model based on conditional variational autoencoder for de novo molecular design. *Journal of
463 cheminformatics*, 10(1):1–9, 2018. 5
- 464 [44] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato,
465 Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel,
466 Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven
467 continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018. 5
- 468 [45] David Weininger. Smiles, a chemical language and information system. 1. introduction to
469 methodology and encoding rules. *Journal of chemical information and computer sciences*, 28
470 (1):31–36, 1988. 5
- 471 [46] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele
472 Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs.
473 *Advances in neural information processing systems*, 33:22118–22133, 2020. 6
- 474 [47] Zhongkai Hao, Chengqiang Lu, Zhenya Huang, Hao Wang, Zheyuan Hu, Qi Liu, Enhong Chen,
475 and Cheekong Lee. Aasn: An active semi-supervised graph neural network for molecular
476 property prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on
477 Knowledge Discovery & Data Mining*, pages 731–752, 2020. 6
- 478 [48] Yao Zhang et al. Bayesian semi-supervised learning for uncertainty-calibrated prediction of
479 molecular properties and active learning. *Chemical science*, 10(35):8154–8163, 2019. 8
- 480 [49] Dávid Bajusz, Anita Rácz, and Károly Héberger. Why is tanimoto index an appropriate choice
481 for fingerprint-based similarity calculations? *Journal of cheminformatics*, 7(1):1–13, 2015. 8
- 482 [50] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A
483 comprehensive survey on graph neural networks. *IEEE transactions on neural networks and
484 learning systems*, 32(1):4–24, 2020. 13
- 485 [51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,
486 Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information
487 processing systems*, pages 5998–6008, 2017. 15
- 488 [52] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann
489 machines. In *ICML*, pages 807–814. Omnipress, 2010. 15
- 490 [53] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint
491 arXiv:1607.06450*, 2016. 15

492 A Appendix

493 A.1 Pool-based batch-mode active learning setting

494 In this setting, we are given an initial pool of labeled data \mathcal{D}_L^0 of size N_L^0 , and an unlabeled pool \mathcal{D}_U^0
 495 of size $N_U^0 > N_L^0$. Our goal is to design an AL algorithm that performs T rounds of querying [7].
 496 In the t -th round ($1 \leq t \leq T$), a batch of b samples, denoted as \mathcal{B}^t , are selected from \mathcal{D}_U^{t-1} . Then,
 497 an oracle (*e.g.*, in our case, a chemist) annotates the queried samples, which are then moved from
 498 the unlabeled pool to the labeled pool. Formally, let $\mathcal{B}_{\text{anno}}^t$ denote annotated selected batch, and then
 499 the pools are updated by $\mathcal{D}_L^t = \mathcal{D}_L^{t-1} \cup \mathcal{B}_{\text{anno}}^t$, and $\mathcal{D}_U^t = \mathcal{D}_U^{t-1} \setminus \mathcal{B}^t$; accordingly $N_L^t = N_L^{t-1} + b$,
 500 $N_U^t = N_U^{t-1} - b$. The obtained \mathcal{D}_L^t is then used to train a *task learner* $h(\cdot)$ — the model that performs
 501 the target learning task at hand (*e.g.*, molecular property prediction in our case). In this paper, the
 502 task learner is a Graph Neural Networks (GNN) [50].

503 Note that, the union of \mathcal{D}_L^t and \mathcal{D}_U^t is always the whole training set, *i.e.*, $\mathcal{D}_{\text{train}} = \mathcal{D}_L^t \cup \mathcal{D}_U^t, \forall t \geq 0$.
 504 Aside from $\mathcal{D}_{\text{train}}$, we also have a validation set \mathcal{D}_{val} and a test set $\mathcal{D}_{\text{test}}$, which are held-out and
 505 disjoint from $\mathcal{D}_{\text{train}}$, for performing model selection and evaluation on the task learner. For brevity,
 506 we omit the round index t unless necessary.

507 A.2 Calculating minimum energy for large n

508 As mentioned in Sec. 4.1.2, we can calculate $\min_{\hat{\mathbf{y}}} E(G, \hat{\mathbf{y}})$ in a task-wise manner, under the
 509 assumption that the n labels are conditionally independent. Here we elaborate on the calculation.

510 In multi-label classification, the n labels are often assumed to be independent given the input. In our
 511 case, this is formulated as: $p(\hat{\mathbf{y}}|G) = \prod_{i=1}^n p(y_i|G)$. Thus, we have

$$\begin{aligned}
 & \arg \min_{\hat{\mathbf{y}}} E(G, \hat{\mathbf{y}}) \\
 &= \arg \max_{\hat{\mathbf{y}}} q(G, \hat{\mathbf{y}}) \\
 &= \arg \max_{\hat{\mathbf{y}}} p(\hat{y}_1, \dots, \hat{y}_n | G) p_L(G) \\
 &= \arg \max_{\hat{\mathbf{y}}} \left(\prod_{i=1}^n p(\hat{y}_i | G) \right) \\
 &= \left(\arg \max_{\hat{y}_1} p(\hat{y}_1 | G), \dots, \arg \max_{\hat{y}_n} p(\hat{y}_n | G) \right).
 \end{aligned} \tag{16}$$

512 This shows that we can calculate $\arg \min_{\hat{\mathbf{y}}} E(G, \hat{\mathbf{y}})$ by simply taking
 513 $(\arg \max_{\hat{y}_1} p(\hat{y}_1 | G), \dots, \arg \max_{\hat{y}_n} p(\hat{y}_n | G))$, without the need of calculating the energy
 514 for all 2^n possible combinations of $(\hat{y}_1, \dots, \hat{y}_n)$.

515 A.3 Energy calculation

516 Our EBM models the score $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ instead of the energy $E(\mathbf{x})$. For obtaining the energy value
 517 $E(\mathbf{x})$, we use summation to approximate integral over $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ after training the EBM.

518 Specifically, the energy of any point \mathbf{x} can be calculated through the following line integral:

$$E(\mathbf{x}) = E(\mathbf{x}_0) + \int_P \nabla_{\mathbf{x}} \log q(\mathbf{x}) \cdot d\mathbf{x}, \tag{17}$$

519 where \cdot denotes a vector inner product. The two terms on the right hand side are explained as follows.

520 The first term $E(\mathbf{x}_0)$ is the energy of an arbitrarily chosen reference point \mathbf{x}_0 . For selecting queries,
 521 we only need the relative energy value. Therefore, without loss of generality, we can take any
 522 reference point \mathbf{x}_0 as the zero-energy point, *i.e.*, letting $E(\mathbf{x}_0) = 0$.

523 The second term is an integral along a path P from \mathbf{x}_0 to \mathbf{x} . Since the true score $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ is a
 524 conservative vector field, the integral result does not depend on the choice of P (assuming that the
 525 EBM approximates $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ well). We thus calculate the integral along the directed line segment
 526 from \mathbf{x}_0 to \mathbf{x} , denoted as $\int_{\mathbf{x}_0}^{\mathbf{x}}$.

527 Finally, the (relative) energy of \mathbf{x} can be calculated by

$$\begin{aligned}
 E(\mathbf{x}) &= \int_{\mathbf{x}_0}^{\mathbf{x}} \nabla_{\mathbf{x}} \log q(\mathbf{x}) \cdot d\mathbf{x} \\
 &\approx \sum_{k=0}^K (\mathbf{x}_{k+1} - \mathbf{x}_k) \cdot \nabla_{\mathbf{x}} \log q(\mathbf{x}_{k+1}) \\
 &= \sum_{k=0}^K (\mathbf{x}_{k+1} - \mathbf{x}_k) \cdot s_{\theta}(\mathbf{x}_{k+1}),
 \end{aligned}
 \tag{18}$$

528 where $\{\mathbf{x}_1, \dots, \mathbf{x}_K\}$ denote K points evenly distributed along the directed line segment, and $\mathbf{x}_{K+1} \triangleq$
 529 \mathbf{x} .

530 A.4 Pseudo codes

531 Alg. 1 and Alg. 2 show the pseudo codes of MMPQ and the model training process, respectively. Notations used can be found in Appx. A.1.

Algorithm 1 Active learning with MMPQ

Input: Initial labeled pool \mathcal{D}_L^0 , unlabeled pool \mathcal{D}_U^0 , number of rounds T , number of queries per round b , EBM, task learner

- 1: Train task learner with \mathcal{D}_L^0 , perform model selection using \mathcal{D}_{val} , and test the learner on $\mathcal{D}_{\text{test}}$
- 2: **for** $t \in \{0, \dots, T\}$ **do**
- 3: Train the EBM (Sec. 4.2.2)
- 4: // perform active selection
- 5: Select a batch of b samples \mathcal{B}^t according to Eqn. (4)
- 6: Annotate \mathcal{B}^t and obtain $\mathcal{B}_{\text{anno}}^t$
- 7: $\mathcal{D}_L^t = \mathcal{D}_L^{t-1} \cup \mathcal{B}_{\text{anno}}^t$, $\mathcal{D}_U^t = \mathcal{D}_U^{t-1} \setminus \mathcal{B}^t$
- 8: // train and test task learner
- 9: Train task learner with \mathcal{D}_L^t , perform model selection using \mathcal{D}_{val} , and test the learner on $\mathcal{D}_{\text{test}}$
- 10: **end for**

Output: Performance on $\mathcal{D}_{\text{test}}$ for $t \in \{0, \dots, T\}$

532

Algorithm 2 Training process

Input: EBM s_{θ} , AE encoder Enc and decoder Dec, and task learner h (trained and fixed)

- 1: **while** not converge **do**
- 2: Randomly sample G from \mathcal{D}_L , G' from \mathcal{D}_U
- 3: Sample $\hat{\mathbf{y}}$ by drawing \hat{y}_i from $\text{Ber}(h(G)_i)$
- 4: $\mathbf{x} = (\text{Enc}(\text{Sml}(G)), \hat{\mathbf{y}})$
- 5: Calculate L_{joint} with Eqn. (9), (10), (11)
- 6: Update s_{θ} , Enc and Dec with L_{joint}
- 7: **end while**

Output: Trained EBM and AE

533 A.5 Dataset information

534 We use the BACE, BBBP, HIV and SIDER datasets from MoleculeNet [5]. Here we give a brief
 535 introduction to these datasets:

- 536 • **BACE:** Human β -secretase 1 (a.k.a. BACE-1) is an enzyme in human body. It is recently found
 537 that inhibition of BACE-1 can slow down the development of Alzheimer’s disease. The BACE
 538 dataset contains experimentally measured binding (*i.e.*, effective in inhibition) results (binarized)
 539 of 1,513 candidate inhibitors of BACE-1.
- 540 • **BBBP:** The Blood–Brain Barrier (BBB) is a highly selective semipermeable border that prevents
 541 solutes in the blood from non-selectively crossing into the extracellular fluid of the central
 542 nervous system. For designing drugs to cure some brain disorder, one major challenge is to
 543 ensure that the obtained drug is able to go through BBB. The BBBP dataset provides binary
 544 labels for 2,039 molecules on their ability to permeate BBB.

- **HIV**: The HIV dataset provides the experimentally tested ability to inhibit HIV replication for 41,127 molecules.
- **SIDER**: SIDER is an abbreviation of Side Effect Resource, which is a database of marketed drugs and their adverse drug reactions (*i.e.*, side effects). The SIDER dataset contains the results of 1,427 drugs on 27 kinds of side effects.

Statistics of the datasets are in Tab. 1.

Table 1: Statistics of used datasets

	BACE	SIDER	BBBP	HIV
Number of Tasks	1	27	1	1
Number of Molecules	1,513	1,427	2,039	41,127

550

551 We note that the TOX21, TOXCAST, MUV and PCBA datasets in the MoleculeNet benchmark [5]
 552 are also often used. However, we find they contain a large number of molecules whose properties not
 553 fully provided by the dataset creator.

554 To see this, for each property of the 4 datasets, we show in Fig. 7 the numbers of molecules whose
 555 ground truth labels are provided or not. We can see that, there are a large number of label-not-provided
 556 molecules for each property that needs to be predicted. This violates the general assumption in AL
 557 that the oracle can correctly annotate each query. Thus we do not run experiments on these datasets.

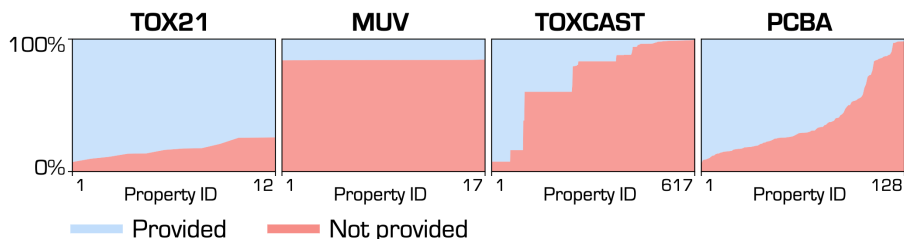


Figure 7: Ratios of molecules whose ground truth properties are provided or not. The x-axis ticks correspond to the index of properties. We sort the properties by the ratio of label-not-provided molecules in ascending order.

558 A.6 Implementation details

559 To obtain reliable results, we randomly sample 50 initial labeled pools in advance, and run each AL
 560 experiment with all these pools and 10 random seeds per pool. The reported performance is averaged
 561 across all initial pools and seeds (*i.e.*, $50 \times 10 = 500$ runs).

562 The AE for reconstructing the SMILES strings is a Transformer [22, 51], which has been shown
 563 to be effective in sequence modeling. The EBM is a 5-layer Multi-Layer Perceptron with residual
 564 connections, ReLU activation function [52], and Layer Normalization [53] between each hidden
 565 layer. We use the Rdkit library¹ to pre-generate SMILES strings, and follow [22] to tokenize the
 566 strings. We instantiate the task learner with a 5-layer GINE architecture [4], which is widely used for
 567 molecular property prediction [4, 36, 37].

568 Before the active learning procedure, the AE is pretrained on the dataset of interest for 5,000 epochs
 569 with learning rate of $5e-4$. In each AL round, the auto-encoder and EBM are jointly trained with batch
 570 size of 128. The learning rates are $2e-4$ for the auto-encoder and $1e-3$ for the EBM. The standard
 571 deviation of the Gaussian noise used in DSM is 10. The criterion for training convergence is that
 572 there are 10 batches whose DSM loss (see Eqn. (10)) is no larger than 0.015, which is obtained in our
 573 pilot experiments. The maximum number of training epochs is 2500. The used optimizer is Adam.
 574 The task learner is trained for 50 epochs with batch size of 128, learning rate of $1e-3$, and the Adam
 575 optimizer. Note that the task learner is re-initialized before it is trained and tested. For calculating the

¹<https://www.rdkit.org/>

576 (relative) energy, we use points for approximation, *i.e.*, $K = 100$ (see Eqn. (18)), and set zero-energy
 577 point as ($\mathbf{z}_0 = \bar{\mathbf{z}}_U, \hat{y}_0 = 0$), where $\bar{\mathbf{z}}_U$ is the mean embedding of the unlabeled pool:

$$\bar{\mathbf{z}}_U = \frac{1}{N_U} \sum_{G \in \mathcal{D}_U} \text{Enc}(\text{Sml}(G)). \quad (19)$$

578 A.7 Numerical results

579 Tab. 2, Tab. 3, Tab. 4 and Tab. 5 show the numerical results (including mean and standard deviation)
 580 of Fig. 2 and Fig. 3. Tab. 6 shows the numerical results of Fig. 4. Tab. 7 shows the numerical results
 581 of Fig. 5.

Table 2: Numerical results on BACE. “Round 0” corresponds to performance on initial labeled pools.

	0	1	2	3	4	5
Random	0.5488 ± .0090	0.5477 ± .0233	0.5556 ± .0243	0.5513 ± .0204	0.5803 ± .0202	0.5975 ± .0196
Entropy	0.5489 ± .0122	0.5469 ± .0172	0.5525 ± .0166	0.5463 ± .0153	0.5566 ± .0161	0.5691 ± .0152
MC-Dropout	0.5510 ± .0110	0.5469 ± .0158	0.5460 ± .0172	0.5413 ± .0163	0.5581 ± .0142	0.5730 ± .0144
CoreSet	0.5497 ± .0113	0.5491 ± .0204	0.5591 ± .0225	0.5563 ± .0212	0.5731 ± .0216	0.5867 ± .0198
ASGN	0.5480 ± .0120	0.5570 ± .0234	0.5634 ± .0217	0.5572 ± .0220	0.5814 ± .0209	0.5934 ± .0219
WAAL	0.5557 ± .0076	0.5699 ± .0141	0.5882 ± .0136	0.6087 ± .0127	0.6108 ± .0155	0.6170 ± .0148
EADA	0.5704 ± .0085	0.5662 ± .0126	0.5685 ± .0101	0.5729 ± .0118	0.5762 ± .0104	0.5827 ± .0099
Badge	0.5501 ± .0118	0.5478 ± .0168	0.5606 ± .0136	0.5569 ± .0157	0.5795 ± .0156	0.5910 ± .0143
MMPQ (Ours)	0.5501 ± .0129	0.5690 ± .0175	0.5935 ± .0183	0.5904 ± .0172	0.6108 ± .0164	0.6160 ± .0152
	6	7	8	9	10	
Random	0.6032 ± .0201	0.6178 ± .0187	0.6246 ± .0190	0.6278 ± .0192	0.6309 ± .0188	
Entropy	0.5749 ± .0149	0.5869 ± .0158	0.5958 ± .0163	0.6012 ± .0142	0.6077 ± .0156	
MC-Dropout	0.5759 ± .0139	0.5846 ± .0132	0.5938 ± .0153	0.5989 ± .0137	0.6025 ± .0139	
CoreSet	0.5885 ± .0186	0.5946 ± .0200	0.6062 ± .0189	0.6084 ± .0176	0.6107 ± .0191	
ASGN	0.5952 ± .0175	0.6099 ± .0192	0.6237 ± .0208	0.6269 ± .0201	0.6355 ± .0188	
WAAL	0.6223 ± .0147	0.6241 ± .0132	0.6244 ± .0130	0.6356 ± .0137	0.6357 ± .0119	
EADA	0.5867 ± .0127	0.5918 ± .0101	0.5955 ± .0102	0.6050 ± .0087	0.6110 ± .0122	
Badge	0.5965 ± .0117	0.6117 ± .0163	0.6191 ± .0151	0.6245 ± .0161	0.6314 ± .0144	
MMPQ (Ours)	0.6176 ± .0133	0.6274 ± .0165	0.6286 ± .0143	0.6337 ± .0146	0.6381 ± .0142	

Table 3: Numerical results on BBBP. “Round 0” corresponds to performance on initial labeled pools.

	0	1	2	3	4	5
Random	0.5944 ± .0099	0.6059 ± .0212	0.6101 ± .0218	0.6214 ± .0213	0.6271 ± .0216	0.6344 ± .0183
Entropy	0.5950 ± .0085	0.6047 ± .0122	0.6055 ± .0156	0.6122 ± .0157	0.6125 ± .0139	0.6176 ± .0116
MC-Dropout	0.5945 ± .0088	0.6035 ± .0130	0.6023 ± .0172	0.6065 ± .0144	0.6071 ± .0153	0.6105 ± .0147
CoreSet	0.5931 ± .0087	0.6036 ± .0199	0.6088 ± .0195	0.6134 ± .0163	0.6169 ± .0182	0.6218 ± .0149
ASGN	0.5936 ± .0074	0.6013 ± .0176	0.6079 ± .0184	0.6231 ± .0161	0.6278 ± .0158	0.6379 ± .0145
WAAL	0.5852 ± .0102	0.5923 ± .0177	0.6076 ± .0169	0.6088 ± .0173	0.6134 ± .0159	0.6183 ± .0164
EADA	0.6014 ± .0075	0.6036 ± .0112	0.6058 ± .0137	0.6073 ± .0130	0.6122 ± .0136	0.6149 ± .0087
Badge	0.5923 ± .0068	0.6030 ± .0086	0.6060 ± .0104	0.6155 ± .0075	0.6191 ± .0092	0.6229 ± .0075
MMPQ (Ours)	0.5943 ± .0075	0.6349 ± .0121	0.6361 ± .0123	0.6389 ± .0126	0.6413 ± .0097	0.6454 ± .0087
	6	7	8	9	10	
Random	0.6373 ± .0198	0.6434 ± .0198	0.6458 ± .0185	0.6509 ± .0190	0.6533 ± .0184	
Entropy	0.6202 ± .0123	0.6237 ± .0135	0.6267 ± .0105	0.6295 ± .0127	0.6312 ± .0125	
MC-Dropout	0.6105 ± .0126	0.6164 ± .0133	0.6191 ± .0143	0.6243 ± .0122	0.6284 ± .0118	
CoreSet	0.6245 ± .0153	0.6268 ± .0129	0.6300 ± .0151	0.6334 ± .0170	0.6363 ± .0166	
ASGN	0.6396 ± .0142	0.6457 ± .0141	0.6470 ± .0145	0.6493 ± .0149	0.6516 ± .0152	
WAAL	0.6231 ± .0144	0.6195 ± .0139	0.6257 ± .0171	0.6253 ± .0168	0.6287 ± .0139	
EADA	0.6163 ± .0103	0.6180 ± .0115	0.6196 ± .0091	0.6211 ± .0095	0.6224 ± .0104	
Badge	0.6234 ± .0091	0.6267 ± .0079	0.6288 ± .0094	0.6350 ± .0077	0.6364 ± .0076	
MMPQ (Ours)	0.6465 ± .0083	0.6490 ± .0087	0.6530 ± .0112	0.6565 ± .0084	0.6585 ± .0082	

Table 4: Numerical results on SIDER. “Round 0” corresponds to performance on initial labeled pools.

	0	1	2	3	4	5
Random	0.5203 ± .0072	0.5248 ± .0165	0.5251 ± .0172	0.5279 ± .0150	0.5309 ± .0155	0.5339 ± .0138
Entropy	0.5205 ± .0081	0.5224 ± .0123	0.5261 ± .0168	0.5270 ± .0103	0.5289 ± .0119	0.5309 ± .0105
MC-Dropout	0.5197 ± .0084	0.5240 ± .0146	0.5255 ± .0144	0.5306 ± .0110	0.5333 ± .0115	0.5364 ± .0124
CoreSet	0.5205 ± .0073	0.5230 ± .0161	0.5271 ± .0184	0.5283 ± .0167	0.5305 ± .0161	0.5346 ± .0145
ASGN	0.5203 ± .0073	0.5217 ± .0163	0.5279 ± .0159	0.5314 ± .0161	0.5332 ± .0148	0.5354 ± .0159
WAAL	0.5262 ± .0064	0.5267 ± .0150	0.5299 ± .0159	0.5300 ± .0144	0.5348 ± .0137	0.5357 ± .0144
EADA	0.5269 ± .0069	0.5270 ± .0091	0.5295 ± .0100	0.5307 ± .0085	0.5337 ± .0104	0.5371 ± .0058
Badge	0.5213 ± .0067	0.5247 ± .0075	0.5258 ± .0100	0.5265 ± .0084	0.5310 ± .0068	0.5339 ± .0080
MMPQ (Ours)	0.5204 ± .0054	0.5246 ± .0087	0.5282 ± .0070	0.5308 ± .0087	0.5363 ± .0068	0.5421 ± .0068
	6	7	8	9	10	
Random	0.5313 ± .0134	0.5363 ± .0137	0.5399 ± .0150	0.5405 ± .0148	0.5463 ± .0164	
Entropy	0.5290 ± .0124	0.5335 ± .0118	0.5369 ± .0107	0.5381 ± .0118	0.5420 ± .0094	
MC-Dropout	0.5356 ± .0145	0.5408 ± .0124	0.5454 ± .0085	0.5455 ± .0105	0.5483 ± .0103	
CoreSet	0.5330 ± .0155	0.5356 ± .0160	0.5388 ± .0148	0.5395 ± .0169	0.5423 ± .0146	
ASGN	0.5339 ± .0136	0.5361 ± .0141	0.5388 ± .0140	0.5397 ± .0130	0.5410 ± .0125	
WAAL	0.5351 ± .0130	0.5380 ± .0147	0.5418 ± .0130	0.5464 ± .0130	0.5473 ± .0121	
EADA	0.5387 ± .0081	0.5405 ± .0080	0.5433 ± .0081	0.5436 ± .0090	0.5449 ± .0085	
Badge	0.5322 ± .0084	0.5363 ± .0080	0.5410 ± .0081	0.5433 ± .0075	0.5468 ± .0070	
MMPQ (Ours)	0.5405 ± .0066	0.5455 ± .0079	0.5494 ± .0066	0.5497 ± .0073	0.5519 ± .0065	

Table 5: Numerical results on HIV. “Round 0” corresponds to performance on initial labeled pools.

	0	1	2	3	4	5
Random	0.6928 ± .0109	0.6928 ± .0241	0.6967 ± .0260	0.6971 ± .0224	0.7015 ± .0244	0.7030 ± .0213
Entropy	0.6949 ± .0098	0.7005 ± .0199	0.7076 ± .0213	0.7094 ± .0158	0.7100 ± .0139	0.7097 ± .0164
MC-Dropout	0.6946 ± .0075	0.7016 ± .0118	0.7063 ± .0122	0.7106 ± .0140	0.7105 ± .0115	0.7123 ± .0114
CoreSet	0.6933 ± .0116	0.6954 ± .0194	0.6996 ± .0234	0.7014 ± .0206	0.7036 ± .0173	0.7037 ± .0200
ASGN	0.6940 ± .0070	0.6999 ± .0173	0.6980 ± .0206	0.6984 ± .0177	0.6989 ± .0169	0.6999 ± .0160
WAAL	0.6587 ± .0121	0.6849 ± .0202	0.6936 ± .0200	0.6973 ± .0200	0.6960 ± .0209	0.6991 ± .0171
EADA	0.6311 ± .0119	0.6558 ± .0170	0.6713 ± .0170	0.6810 ± .0175	0.6865 ± .0175	0.6899 ± .0145
Badge	0.6958 ± .0086	0.7019 ± .0119	0.7054 ± .0150	0.7080 ± .0095	0.7096 ± .0103	0.7125 ± .0097
MMPQ (Ours)	0.6954 ± .0073	0.7198 ± .0121	0.7262 ± .0125	0.7302 ± .0123	0.7302 ± .0103	0.7293 ± .0112
	6	7	8	9	10	
Random	0.7065 ± .0184	0.7107 ± .0198	0.7123 ± .0177	0.7144 ± .0211	0.7165 ± .0216	
Entropy	0.7122 ± .0184	0.7135 ± .0152	0.7146 ± .0152	0.7161 ± .0154	0.7190 ± .0159	
MC-Dropout	0.7128 ± .0132	0.7142 ± .0124	0.7142 ± .0126	0.7161 ± .0115	0.7164 ± .0109	
CoreSet	0.7054 ± .0177	0.7072 ± .0170	0.7090 ± .0193	0.7111 ± .0188	0.7110 ± .0164	
ASGN	0.7060 ± .0162	0.7101 ± .0155	0.7155 ± .0138	0.7196 ± .0119	0.7225 ± .0152	
WAAL	0.7001 ± .0199	0.7019 ± .0159	0.7021 ± .0185	0.7063 ± .0166	0.7055 ± .0183	
EADA	0.6929 ± .0146	0.6943 ± .0136	0.6957 ± .0122	0.6971 ± .0155	0.6970 ± .0157	
Badge	0.7143 ± .0103	0.7153 ± .0091	0.7153 ± .0092	0.7155 ± .0107	0.7159 ± .0100	
MMPQ (Ours)	0.7294 ± .0113	0.7291 ± .0109	0.7294 ± .0103	0.7300 ± .0099	0.7302 ± .0077	

Table 6: Numerical results of Fig. 4 (a). “Round 0” corresponds to performance on initial labeled pools.

	0	1	2	3	4	5
MMPQ	0.6954 ± .0073	0.7198 ± .0121	0.7262 ± .0125	0.7302 ± .0123	0.7302 ± .0103	0.7293 ± .0112
U.O.	0.6943 ± .0077	0.7074 ± .0116	0.7126 ± .0127	0.7172 ± .0111	0.7208 ± .0113	0.7216 ± .0102
D.O.	0.6947 ± .0069	0.7218 ± .0104	0.7192 ± .0110	0.7197 ± .0088	0.7193 ± .0085	0.7192 ± .0106
	6	7	8	9	10	
MMPQ	0.7294 ± .0113	0.7291 ± .0109	0.7294 ± .0103	0.7300 ± .0099	0.7302 ± .0077	
U.O.	0.7239 ± .0075	0.7269 ± .0084	0.7274 ± .0090	0.7282 ± .0091	0.7288 ± .0098	
D.O.	0.7191 ± .0127	0.7184 ± .0106	0.7177 ± .0097	0.7169 ± .0112	0.7160 ± .0105	

Table 7: Numerical results of Fig. 5. “Round 0” corresponds to performance on initial labeled pools.

	0	1	2	3	4	5
$(\bar{z}_L, 0, 100)$	0.6922 ± .0089	0.7176 ± .0115	0.7252 ± .0153	0.7291 ± .0103	0.7292 ± .0121	0.7291 ± .0138
$(\bar{z}_F, 0, 100)$	0.6941 ± .0097	0.7189 ± .0115	0.7245 ± .0151	0.7293 ± .0141	0.7296 ± .0132	0.7299 ± .0126
$(\bar{z}_U, 0, 50)$	0.6942 ± .0085	0.7217 ± .0108	0.7282 ± .0106	0.7283 ± .0106	0.7285 ± .0111	0.7292 ± .0110
$(\bar{z}_U, 0, 500)$	0.6940 ± .0096	0.7193 ± .0140	0.7248 ± .0162	0.7287 ± .0113	0.7281 ± .0125	0.7294 ± .0123
$(\bar{z}_U, 1, 100)$	0.6923 ± .0105	0.7208 ± .0119	0.7292 ± .0138	0.7301 ± .0126	0.7298 ± .0124	0.7293 ± .0124
	6	7	8	9	10	
$(\bar{z}_L, 0, 100)$	0.7299 ± .0150	0.7295 ± .0126	0.7291 ± .0121	0.7298 ± .0092	0.7294 ± .0133	
$(\bar{z}_F, 0, 100)$	0.7290 ± .0125	0.7299 ± .0124	0.7293 ± .0106	0.7294 ± .0122	0.7291 ± .0122	
$(\bar{z}_U, 0, 50)$	0.7290 ± .0091	0.7294 ± .0093	0.7293 ± .0110	0.7294 ± .0092	0.7294 ± .0091	
$(\bar{z}_U, 0, 500)$	0.7298 ± .0124	0.7294 ± .0115	0.7292 ± .0137	0.7298 ± .0119	0.7295 ± .0102	
$(\bar{z}_U, 1, 100)$	0.7299 ± .0119	0.7287 ± .0105	0.7297 ± .0111	0.7300 ± .0108	0.7305 ± .0114	

582 A.8 Generalization to categorical classification or regression

583 Recall that, in MMPQ, we use $\max_{\hat{y}} p(\hat{y}|G)$ as a metric of model uncertainty. It is easy to see that,
 584 in the binary classification case, selecting those with small $\max_{\hat{y}} p(\hat{y}|G)$ is equivalent to selecting
 585 those with a large prediction entropy, which is a widely-used uncertainty metric. However, this may
 586 not hold in categorical classification or regression tasks.

587 We show in this subsection that, by choosing the KL-divergence between the prediction distribution
 588 and the uniform distribution as the uncertainty metric, we can generalize our approach to categorical
 589 classification or regression tasks. For simplicity, we take single-task categorical classification as an
 590 example.

591 We start with categorical classification. Suppose that there are C classes in total, and denote $p_c =$
 592 $p(\hat{y} = c|G)$ ($c \in \mathcal{C} \triangleq \{1, \dots, C\}$). The KL-divergence between the prediction distribution $p(\hat{y}|G)$
 593 and a uniform distribution is

$$\begin{aligned}
 D_{\text{KL}} &= \sum_{c=1}^C p_c \log\left(\frac{p_c}{\frac{1}{C}}\right) \\
 &= \sum_{c=1}^C p_c (\log(p_c) + \log(C)) \\
 &= \log(C) - \left(-\sum_{c=1}^C p_c \log(p_c)\right) \\
 &= \log(C) - H(p),
 \end{aligned} \tag{20}$$

594 where $H(p)$ is the entropy of the prediction distribution.

595 Since $\log(C)$ is a constant, selecting those with a small D_{KL} is equivalent to selecting those with a
 596 large $H(p)$. This shows that, D_{KL} is equivalent to $H(p)$ in terms of measuring model uncertainty.

597 Then, by replacing $\max_{\hat{y}} p(\hat{y}|G)$ with $H(p)$ in Eqn. 5, we obtain a new selection score:

$$\begin{aligned}
 H(p)p_L(G) &= - \left(\sum_{c=1}^C p_c \log(p_c) \right) p_L(G) \\
 &= - \sum_{\hat{y} \in \mathcal{C}} q(G, \hat{y}) \log(p(\hat{y}|G)) \\
 &= - \sum_{\hat{y} \in \mathcal{C}} q(G, \hat{y}) \log \left(\frac{q(G, \hat{y})}{\sum_{\hat{y}' \in \mathcal{C}} q(G, \hat{y}')} \right) \\
 &\propto - \sum_{\hat{y} \in \mathcal{C}} \exp(-E(G, \hat{y})) \log \left(\frac{\exp(-E(G, \hat{y}))}{\sum_{\hat{y}' \in \mathcal{C}} \exp(-E(G, \hat{y}'))} \right).
 \end{aligned} \tag{21}$$

598 The above equation shows that the selection score can also be computed using the EBM.

599 With the same reasoning, the selection score for regression tasks is:

$$- \int_{\hat{y} \in Y} \exp(-E(G, \hat{y})) \log \left(\frac{\exp(-E(G, \hat{y}))}{\int_{\hat{y}' \in Y} \exp(-E(G, \hat{y}')) d\hat{y}'} \right) d\hat{y}, \tag{22}$$

600 where Y is the support set.

601 Note that, the above formulation requires computing the integration over Y , which is intractable.

602 This might be approximated by sampling and summation.

603 **A.9 Main results with unified y-axis scale**

604 For better comparison, the Fig. 8 and Fig. 9 show the results of Fig. 2 and Fig. 3 with the same y-axis scale for each dataset.

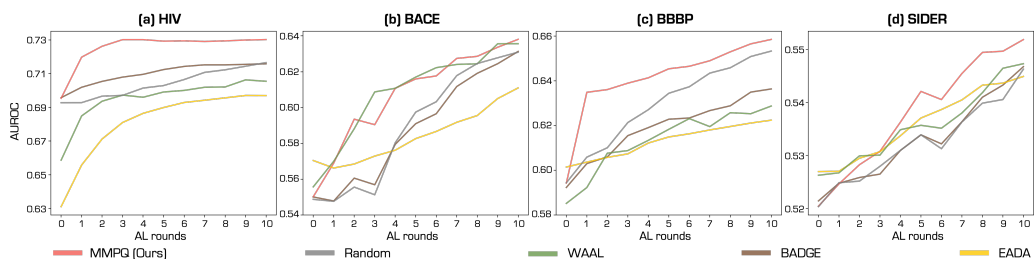


Figure 8: Active learning performance of MMPQ (ours) and baseline hybrid methods. “Round 0” corresponds to the performance on initial labeled pool.

605

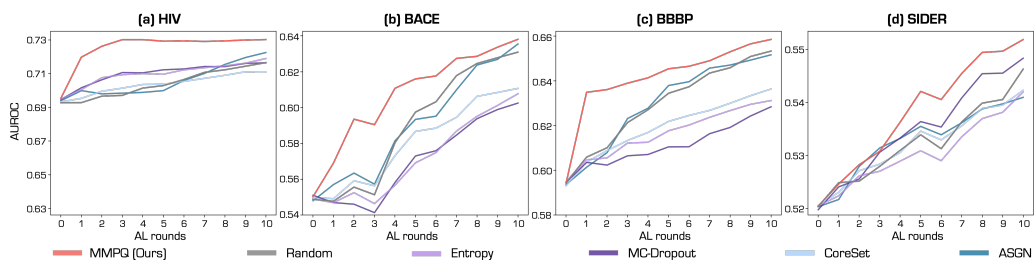


Figure 9: Active learning performance of MMPQ (ours) and uncertainty-based or diversity-based methods. “Round 0” corresponds to the performance on initial labeled pool.