Message-Passing Monte Carlo: Generating low-discrepancy point sets via Graph Neural Networks

T. Konstantin Rusch[a,1], **Nathan Kirk**[b], **Michael M. Bronstein**[c], **Christiane Lemieux**[b], **and Daniela Rus**[a]

Discrepancy is a well-known measure for the irregularity of the distribution of a point set. Point sets with small discrepancy are called low-discrepancy and are known to efficiently fill the space in a uniform manner. Low-discrepancy points play a central role in many problems in science and engineering, including numerical integration, computer vision, machine perception, computer graphics, machine learning, and simulation. In this work, we present the first machine learning approach to generate a new class of low-discrepancy point sets named *Message-Passing Monte Carlo (MPMC)* points. Motivated by the geometric nature of generating low-discrepancy point sets, we leverage tools from Geometric Deep Learning and base our model on Graph Neural Networks. We further provide an extension of our framework to higher dimensions, which flexibly allows the generation of custom-made points that emphasize the uniformity in specific dimensions that are primarily important for the particular problem at hand. Finally, we demonstrate that our proposed model achieves state-of-the-art performance superior to previous methods by a significant margin. In fact, MPMC points are empirically shown to be either optimal or near-optimal with respect to the discrepancy for low dimension and small number of points, i.e., for which the optimal discrepancy can be determined. Code for generating MPMC points can be found at https://github.com/tk-rusch/MPMC

Low-discrepancy, Machine Learning, Graph Neural Networks, Geometric Deep Learning, Quasi-Monte Carlo

Significance Statement

This article introduces *Message-Passing Monte Carlo (MPMC)*, the first machine learning approach for generating low-discrepancy point sets which are essential for efficiently filling space in a uniform manner, and thus play a central role in many problems in science and engineering. To accomplish this, MPMC utilizes tools from Geometric Deep Learning, specifically by employing Graph Neural Networks. MPMC can be extended to a higher-dimensional case which further allows for generating custom-made points. Finally, MPMC point sets significantly outperform previous methods, achieving near-optimal discrepancy in practice for low dimension and small number of points, i.e., for which the optimal discrepancy can be determined. This advancement holds promise for enhancing efficiency in fields like scientific computing, computer vision, machine learning, and simulation.

Monte Carlo (MC) methods have been commonly used and are a popular choice for approximating and simulating complex real-world systems. Known for their reliance on repeated random sampling, MC methods function well in problems involving optimization, numerical integration, and financial mathematics (particularly derivative pricing and risk management) via computer simulation. However, their convergence rate of $\mathcal{O}(N^{-1/2})$ in the number of samples N means that achieving high precision with MC requires an impractically large number of samples for complex problems. To address this drawback, it is common to employ *variance reduction techniques* such as importance sampling, stratified sampling, or control variates to obtain the same degree of accuracy with fewer samples (for details, see (1), (2) and references therein).

A particularly successful approach for convergence is called *quasi-Monte Carlo* (QMC). QMC methods employ a deterministic point set, which replaces the purely random sampling with a sample whose points span the hypercube $[0,1]^d$ in a manner that is more uniform than what can be achieved with MC sampling. The fact that these point sets are constructed over $[0,1]^d$ is not overly restrictive as most, if not all, sampling algorithms used within the MC method take as input (pseudo)random numbers in $[0,1]$. The uniformity of these deterministic point sets (or indeed, any point set) can be captured by one of several of measures of irregularity of distribution, referred to by the umbrella term *discrepancy measures* (3). The more uniformly distributed the points are, the lower the discrepancy is; point sets possessing a small enough discrepancy value are called *low-discrepancy*. In the classical setting, the *star-discrepancy*, widely regarded as the most important uniformity measure, of an $N-$element point set $\{\mathbf{X}_i\}_{i=1}^N$ contained in $[0,1]^d$ represents the largest absolute difference between the volume of a test box and the proportion of points of $\{\mathbf{X}_i\}_{i=1}^N$ that fall inside the test box,

$$D^*\left(\{\mathbf{X}_i\}_{i=1}^N\right) := \sup_{\mathbf{x}\in[0,1]^d} \left| \frac{\#\left(\{\mathbf{X}_i\}_{i=1}^N \cap [0,\mathbf{x})\right)}{N} - \mu([0,\mathbf{x})) \right| \qquad [1]$$

where $\#(\{\mathbf{X}_i\}_{i=1}^N \cap [0,\mathbf{x}))$ counts how many points of $\{\mathbf{X}_i\}_{i=1}^N$ fall inside the box $[0,\mathbf{x}) = \prod_{i=1}^d [0,x_i)$ for $\mathbf{x} = (x_1,\dots,x_d) \in [0,1]^d$, and $\mu(\cdot)$ denotes the usual

Author affiliations: [a]Massachusetts Institute of Technology (MIT); [b]University of Waterloo; [c]University of Oxford

[1]To whom correspondence should be addressed. E-mail: tkrusch@mit.edu

Lebesgue measure. The discrepancy is closely related to worst-case integration error of a particular class of functions with the most well-known result being the Koksma-Hlawka inequality; see (4, 5). Explicitly, given a point set $\{\mathbf{X}_i\}_{i=1}^N$ contained in $[0,1]^d$, we have

$$\left| \int_{[0,1]^d} f(\mathbf{x}) d\mathbf{x} - \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{X}_i) \right| \leq D^*(\{\mathbf{X}_i\}_{i=1}^N) V(f) \quad [2]$$

where $V(f)$ denotes the variation of the function f in the sense of Hardy and Krause. This result illustrates that points with small discrepancy induce approximations with small errors. Thus in summary, it is of general interest to find $N-$point configurations with smallest discrepancy; see (6–10) for examples of QMC implementation.

Given this context, our main goal is to present a machine learning framework that generates point sets with minimal discrepancy. Based on the geometric nature of this problem, we suggest to leverage graph-learning models from Geometric Deep Learning (11) to achieve this. More concretely, we construct a computational graph based on nearest neighbors of the initial input points and process the encoded input points with a deep message-passing neural network, which is trained to minimize a closed-form solution of a specific discrepancy measure of its decoded and clamped outputs. We term the resulting low-discrepancy points **Message-Passing Monte Carlo (MPMC)** points. Previous methods either fail to achieve optimal discrepancy values or are computationally intractable, being limited to small dimensions ($d \leq 3$) and very small numbers of points ($N \leq 21$), which can still require weeks of computation. In contrast, MPMC can be trained in a few minutes to achieve near-optimal discrepancy for these cases. Moreover, we show that MPMC is not limited to small number of points in small dimensions but can efficiently generate $N > 1000$ points in tens of dimensions. This advancement represents a significant step forward in the development of highly efficient sampling methods, which are crucial for many applications in science and engineering. Concrete examples include problems in financial mathematics (12), path and motion planning in robotics (13), and enhanced training of neural scene rendering methods like Neural Radiance Fields (NeRFs) (14).

Main contributions. In the subsequent sections, we will:

- introduce a new state-of-the-art machine learning model that generates low-discrepancy points. To our knowledge, this is the first machine learning approach in this context.
- extend our framework to higher dimensions by minimizing the average discrepancy of randomly selected subsets of projections. This allows for generating custom-made points that emphasize specific dimensions that are primarily important for the particular problem at hand.
- provide an extensive empirical evaluation of our proposed MPMC point sets and demonstrate their superior performance over previous methods.

1. Background and previous work

Our general goal in this paper is to provide a method for generating point sets with small discrepancy. In the following, we use the term *sequence* to refer to an infinite series of points, and *point set* for a finite one.

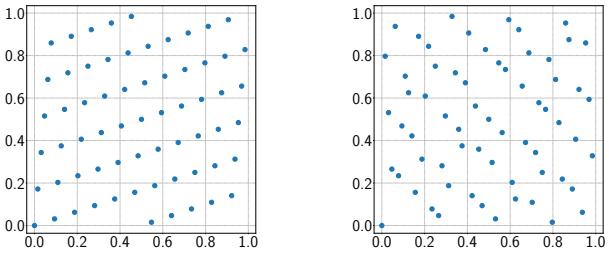

Fig. 1. Two different low-discrepancy point sets with $N = 64$: Korobov lattice (left), and Sobol' (right).

A sequence of points $\{\mathbf{X}_i\}_{i=1}^{\infty}$ contained in $[0,1]^d$ is called a low-discrepancy sequence if the star-discrepancy of the first N points satisfies $D^*\left(\{\mathbf{X}_i\}_{i=1}^N\right) = \mathcal{O}((\log N)^d/N)$. A finite point set $\{\mathbf{X}_i\}_{i=1}^N$ is said to be of low discrepancy if its corresponding star-discrepancy $D^*\left(\{\mathbf{X}_i\}_{i=1}^N\right)$ is "small" enough, which in practice means that a bound of the form $c(\log N)^{d-1}/N$ can be established, for a given constant c independent of N (but possibly dependent on d). Moving forward, for comparison purposes, we will truncate various known infinite low-discrepancy sequences resulting in a finite point set, which inherits the low-discrepancy property from the underlying infinite sequence.

Figure 1 illustrates two examples of low-discrepancy point sets. On the left-hand side we have a Korobov lattice (15), which is an example of a lattice rule (16–19), and on the right-hand side, we see the first 64 points of the two-dimensional Sobol' sequence (20). This construction leverages a widely used building block for many low-discrepancy sequences known as the van der Corput sequence in base b (21). It is also an example of what are modernly known as digital (t,s)-sequences—which also include the Faure sequences (22)—that were first laid out in (23), with a comprehensive overview provided in the subsequent monograph (24). Halton sequences (25) are another widely used type of low-discrepancy sequences that concatenate d van der Corput sequences in different bases, usually taken as the first d prime numbers.

More recently, there have been successful attempts to construct low-discrepancy point sets using more sophisticated means motivated by the lack of constructions adapted to specific N and d. In (26), new low-discrepancy point sets were suggested via the optimization of permutations applied to a Halton sequence. More recently, a method called *subset selection* is formulated to choose from an $N-$element point set, the $k < N$ points which yield the smallest discrepancy. An exact selection algorithm was presented in (27), while a swap-based heuristic approach was used in (28). Furthermore, a method to generate optimal star-discrepancy point sets for fixed N and d based on a non-linear programming approach was suggested in (29). However, this formulation of the problem presented huge computational burdens allowing optimal sets only to be found up to 21 points in dimension two and 8 points in dimension three.

2. Method

Let $1 < d < +\infty$ and $1 \leq N < +\infty$ be fixed natural numbers. Our objective is to train a neural network to transform

(random) input points $\{\mathbf{X}_i\}_{i=1}^N$ into points $\{\hat{\mathbf{X}}_i\}_{i=1}^N$ that reduce the star-discrepancy D^* Eq. (1), where $\mathbf{X}_i, \hat{\mathbf{X}}_i \in [0,1]^d$ for all i.

In this work, we propose to leverage Graph Neural Networks (GNNs) (30–39) based on the message-passing framework to effectively learn such transformations. GNNs are a popular class of model architectures for learning on relational data, and have successfully been applied on a variety of different tasks, e.g., in computer science (39–41), and the natural sciences (42–44) (see (11, 45) for additional applications). In particular, GNNs have successfully been used in the context of learning on point clouds, or generally learning on sets. This motivates the choice of GNNs in our setup, where specific transformations of geometric sets (i.e., set of input points in $[0,1]^d$) have to be learned.

A schematic drawing of our approach can be seen in Fig. 2, where we train a GNN model to transform $N = 64$ random input points $\{\mathbf{X}_i\}_{i=1}^N$ into low-discrepancy points $\{\hat{\mathbf{X}}_i\}_{i=1}^N$.

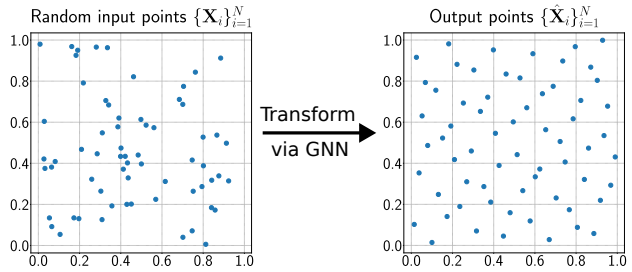

Fig. 2. Schematic drawing of our proposed approach to transform (random) input points $\{\mathbf{X}_i\}_{i=1}^N$ into low-discrepancy points $\{\hat{\mathbf{X}}_i\}_{i=1}^N$. Both the input and output point sets are actual instances of our proposed model, with $N = 64$ and $d = 2$ in this example.

2.A. Training set. Our approach can be classified as an unsupervised learning setup, where, in contrast to supervised learning, only input data is required without any labels. While it is intuitive to generate the set of input points randomly, we suggest several different approaches for constructing input data. First, using aforementioned uniform random sampled set of input points $\mathbf{X}_i \sim \mathcal{U}([0,1]^d)$, for all points $i = 1, \dots, N$. Second, using set of input points from available low-discrepancy constructions, such as Sobol', Halton, or a lattice rule. Third, using set of input points from randomly perturbed low-discrepancy points, i.e.,

$$\mathbf{X}_i = \mathbf{Y}_i + \xi \pmod 1, \qquad [3]$$

where $\mathbf{Y}_i$ is generated by a known low-discrepancy set and ξ is uniform randomly sampled from $[0,b]^d$, with $0 < b \leq 1$, for all points $i = 1, \dots, N$.

2.B. Model architecture. We start by constructing an undirected computational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V})$, where $\mathcal{V}$ denotes the set of unordered nodes corresponding to the input points $\{\mathbf{X}_i\}_{i=1}^N$, and $\mathcal{E}$ is the set of pair-wise connections between the nodes. In addition, each node $i \in \mathcal{V}$ is equipped with a node feature set to the coordinates of an input point, i.e., set to $\mathbf{X}_i \in \mathbb{R}^d$. We further denote the 1-*neighborhood* of a node $i \in \mathcal{V}$ as $\mathcal{N}_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}$. Clearly, the set of all 1-neighborhoods induces the connectivity of the graph, i.e., the set of node-wise connections $\mathcal{E}$. Hence, the only remaining part of the construction of the underlying computational graph $\mathcal{G}$ is to define the local structure of the graph, i.e., defining $\mathcal{N}_i$ for all nodes i. It is worth noting that in many GNN applications (e.g., network science, or life sciences) the computational graph structure is already given a-priori, either explicitly or implicitly. In contrast to that, our problem setup considers the connectivity of the underlying computational graph as an additional design choice. While there are many suitable choices, often balancing a global vs local connectivity structure, we suggest basing the local connectivity on the nearest neighbors of the node features, i.e., for a fixed radius $0 < r \leq \sqrt{d}$,

$$\mathcal{N}_i = \{j \in \mathcal{V} : \|\mathbf{X}_i - \mathbf{X}_j\|_2 \leq r\}. \qquad [4]$$

We choose this inherently local structure to guide the GNN training towards transforming input points into low-discrepancy points by mainly considering the positions of other near-by points (in the corresponding Euclidean space of the input point set).

We can now define the main building block of our MPMC model, i.e., GNN layers based on the message-passing framework. Message-passing GNNs are a family of parametric functions defined through local updates of hidden node representations. More concretely, we iteratively update node features as,

$$\mathbf{X}_i^l = \phi^l\left(\mathbf{X}_i^{l-1}, \sum_{j \in \mathcal{N}_i} \psi^l(\mathbf{X}_i^{l-1}, \mathbf{X}_j^{l-1})\right), \text{for all } l = 1, \dots, L, \qquad [5]$$

with $\mathbf{X}_i^l \in \mathbb{R}^{m_l}$ for all nodes i. Moreover, we parameterize ϕ^l, ψ^l as ReLU-multilayer perceptrons (MLPs), i.e., MLPs using the element-wise $\text{ReLU}(x) = \max(0,x)$ activation function in-between layers. We further encode the initial node features by an affine transformation $\mathbf{X}_i^0 = \mathbf{A}_{\text{enc}}\mathbf{X}_i + \mathbf{b}_{\text{enc}}$ for all $i = 1, \dots, N$, with weight matrix $\mathbf{A}_{\text{enc}} \in \mathbb{R}^{m_0 \times d}$ and bias $\mathbf{b}_{\text{enc}} \in \mathbb{R}^{m_0}$. Finally, we decode the output of the final GNN layer by an affine transformation and smoothly clamp the decoded outputs back into $[0,1]^d$ by using the element-wise sigmoidal activation function, i.e., $\hat{\mathbf{X}}_i = \sigma(\mathbf{A}_{\text{dec}}\mathbf{X}_i^L + \mathbf{b}_{\text{dec}})$ for all $i = 1, \dots, N$, with sigmoidal function $\sigma(x) = 1/(1+e^{-x})$, weight matrix $\mathbf{A}_{\text{dec}} \in \mathbb{R}^{d \times m_L}$, and bias $\mathbf{b}_{\text{dec}} \in \mathbb{R}^d$. Note that clamping is crucial, as otherwise the training objectives we introduce in the subsequent sections are ill-defined. A schematic of the full model can be seen in Fig. 3.

2.C. Training objective. Our ultimate goal is to minimize the star-discrepancy D^* Eq. (1). However, D^* cannot serve as the training objective, as (i) D^* is computationally infeasible to calculate for high dimensions d and large number of points N, (ii) the training objective should not only be computationally feasible but rather very efficient to compute, as it needs to be evaluated at every step of the training procedure (i.e., for every step of the gradient descent method) resulting in potentially thousands of evaluations to train only a single model, and (iii) the training objective needs to be sufficiently differentiable (i.e., that can be handled by automatic differentiation packages such as (46)) in order to be used in the context of gradient-based learning. It turns out, we can derive a training objective resolving all three issues

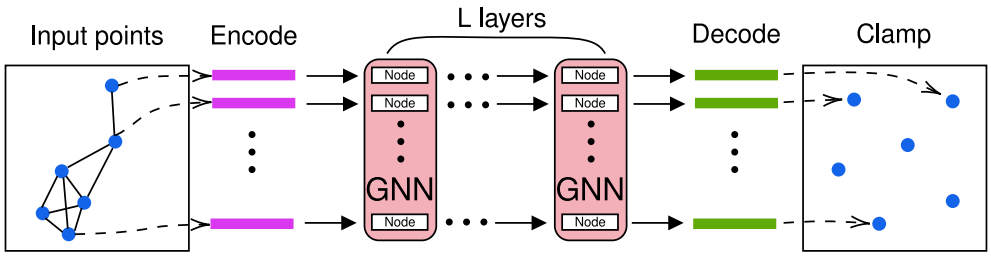

Fig. 3. Schematic of the proposed model to learn low-discrepancy points. First, (random) input points $\{\mathbf{X}_i\}_{i=1}^N$ are encoded to a high dimensional representation. Second, the encoded representations are passed through a deep GNN Eq. (5), where the underlying computational graph is constructed based on nearest neighbors using the positions of the initial input points. Finally, the node-wise output representations of the final GNN layer are decoded and clamped yielding new d-dimensional points $\{\hat{\mathbf{X}}_i\}_{i=1}^N$ in $[0,1]^d$.

while simultaneously reducing D^* by leveraging previous work on the $\mathcal{L}_p$-discrepancy,

$$\mathcal{L}_p(\{\mathbf{X}_i\}_{i=1}^N) := \left(\int_{[0,1]^d} \left|\frac{\#(\{\mathbf{X}_i\}_{i=1}^N \cap [0,\mathbf{x}))}{N} - \mu([0,\mathbf{x}))\right|^p d\mathbf{x}\right)^{\frac{1}{p}}. \quad [6]$$

Clearly, the star-discrepancy D^* can be derived as a special case of Eq. (6) with $p=\infty$. Here, we focus on the case of $p=2$ as our training objective, since instead of computing the integral in Eq. (6), we can leverage its closed-form expression, known as Warnock's formula (47),

$$\mathcal{L}_2^2(\{\mathbf{X}_i\}_{i=1}^N) = \frac{1}{3^d} - \frac{2}{N}\sum_{i=0}^{N-1}\prod_{k=0}^{d}\frac{1-\mathbf{X}_{i,k}^2}{2} + \frac{1}{N^2}\sum_{i,j=0}^{N-1}\prod_{k=0}^{d} 1-\max(\mathbf{X}_{i,k},\mathbf{X}_{j,k}), \quad [7]$$

where $\mathbf{X}_{i,k}$ is the k-th entry of $\mathbf{X}_i$. This enables a very fast and exact computation of the $\mathcal{L}_2$-discrepancy without errors resulting from numerical quadrature methods. Thus, the $\mathcal{L}_2$-discrepancy is an ideal candidate for the training objective of our machine learning approach.

2.D. Extension to higher dimensions. In many practical problems, particularly in engineering and finance, the dimension d of the problem can be very large. This necessitates extending low-discrepancy sequences to the high dimensional case of $d \gg 1$. However, it is known (48, 49) that the $\mathcal{L}_2$-discrepancy fails to identify superior distributional properties of low-discrepancy point sets over random samples as the dimension increases. Indeed, in high dimensions the classical $\mathcal{L}_2$-discrepancy of low-discrepancy point sets behaves like $\mathcal{O}(1/\sqrt{N})$, the same as for random points, for moderate values of N, while an improved order close to $\mathcal{O}(1/N)$ can only be seen for extremely large N. Empirical evidence for these last claims can be found in the discrepancy plots contained in (48).

To this end, we suggest to base our new training objective for higher-dimensional generation of low-discrepancy points on the Hickernell $\mathcal{L}_p$-discrepancy (50),

$$D_{H,p}(\{\mathbf{X}_i\}_{i=1}^N) = \left(\sum_{\emptyset\neq s\subseteq\{1,\dots,d\}} \mathcal{L}_p^p(\{\mathbf{X}_i^s\}_{i=1}^N)\right)^{\frac{1}{p}}, \quad [8]$$

where $\emptyset \neq s \subseteq \{1,\dots,d\}$ is a non-empty subset of coordinate indices, and $\{\mathbf{X}_i^s\}_{i=1}^N$ is the projection of $\{\mathbf{X}_i\}_{i=1}^N$ onto $[0,1]^{|s|}$.

Note that while we can again make use of Warnock's formula Eq. (7) to compute $D_{H,2}$, it requires computing the sum of the $\mathcal{L}_2$-discrepancy of 2^d-1 projections, which already for $d=32$ is more than 1B. This highlights the necessity of modifying $D_{H,2}$ in order for it to be used as a training objective in a machine learning framework. Therefore, we suggest to base the training objective on a modification of the Hickernell $\mathcal{L}_p$-discrepancy via random projections,

$$\tilde{D}_{H,p,K}(\{\mathbf{X}_i\}_{i=1}^N) = \left(\sum_{k=1}^{K}\mathcal{L}_p^p(\{\mathbf{X}_i^{s_k}\}_{i=1}^N)\right)^{\frac{1}{p}}, \quad [9]$$

where $\emptyset \neq s_k \sim \mathcal{P}(\{1,\dots,d\})$ are randomly sampled subsets of coordinate indices for each $k=1,\dots,K$, thus requiring to compute the $\mathcal{L}_p$-discrepancy only K times. More specifically, we sample s_k by first selecting the dimension d_k of the projection uniformly at random, and then sampling the projection itself uniformly at random from the set of all d_k-dimensional projections. This method ensures that the dimensions of the projections are uniformly distributed during training. Alternatively, we could randomly sample the dimensions of the projections according to a binomial distribution.

Generating problem-dependent point sets. As a further advantage to this framework, we highlight its inherent flexibility. Specifically, employing the modified Hickernell discrepancy as the training objective represents a *first step towards an adaptive QMC sampling method tailored for specific problems*. It is widely recognized that for many problems, the effective dimension—essentially, the number of dimensions capturing the majority of the problem's variability—is often significantly lower than the nominal dimension; for full details, refer to (51). Therefore, during high-dimensional training, assuming that the important subsets of variables are known or identified in advance, e.g., by functional ANOVA methods (52), prioritizing sampling from specific lower dimensional projections will yield a d-dimensional point set that is highly uniformly distributed in those same projections identified during training. This approach effectively creates a custom-made point set, optimized for problems that primarily depend upon particular subsets of variables.

3. Empirical results

In this section, we present empirical results comparing MPMC points to current state-of-the-art low-discrepancy point sets. More concretely, we demonstrate superior distributional

properties of MPMC over other low-discrepancy point sets mainly with respect to the star-discrepancy, D^*.

As previously mentioned, computing the star-discrepancy is typically a challenging task. The *DEM algorithm* (53) denotes the fastest method to compute the exact star-discrepancy with a significantly reduced complexity, running in $\mathcal{O}(N^{1+d/2})$ time. In all experiments, the results of which are presented shortly, to calculate the star discrepancy we either use a simple crude search or a parallelized version of the DEM algorithm from (54) to speed up calculation when necessary. The interested reader is recommended to consult (55) and references therein for more information on the calculation of the star-discrepancy, and indeed the computation of other discrepancy measures.

3.A. Low-dimensional generation of MPMC points. Here, we focus on generating MPMC points within a lower-dimensional setting particularly because this area has recently attracted significant attention (27–29) providing a solid basis for comparison.

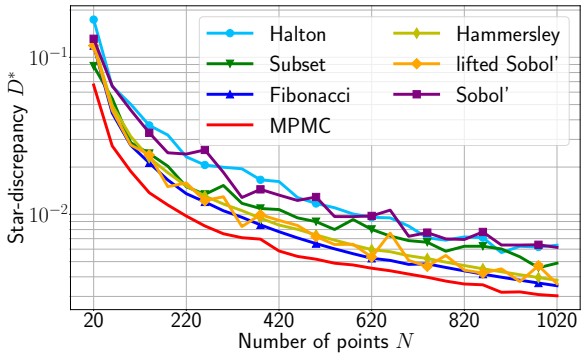

Fig. 4. Star-discrepancy D^* of Halton, Sobol', lifted Sobol', Subset Selection, Hammersley, Fibonacci, and MPMC for increasing number of points $N = 20, \ldots, 1020$ in $d = 2$.

We compare the irregularity of our MPMC point sets with a truncation to the first N points of the widely used Sobol' and Halton sequences. We note that MPMC points are sets optimized for N chosen in advance, whereas a key advantage of the Sobol' and Halton sequences are that they are built for repeated sampling and retain a low-discrepancy for all values of N. Therefore, in addition, we provide comparison with state-of-the-art point sets derived from the subset selection method from (27, 28), lifted Sobol' (the first N terms of a one-dimensional Sobol' sequence concatenated with i/N for $i \in \{0, \ldots, N-1\}$), the Hammersley construction in base $1+\sqrt{2}$ as introduced in (56) and the Fibonacci set defined as $\{(i/N, \{i\varphi\}) : i \in \{0, \ldots, N-1\}\}$ where φ represents the golden ratio, the notation $\{x\}$ denotes the fractional part of $x \in \mathbb{R}$. All four of these sets are recognized for having among the lowest star-discrepancy for given N in two dimensions. Fig. 4 shows the star-discrepancy of MPMC, Sobol', Halton, subset selection, lifted Sobol', Hammersley and Fibonacci sets in two dimensions for increasing number of points $N = 20, \ldots, 1020$. We can see that MPMC significantly outperforms all other methods with respect to the star-discrepancy. In fact, the star-discrepancy of MPMC is on average 1.3 times smaller than that of the current state-of-the-art Fibonacci construction, and on average more than 2.2 times smaller than Sobol' or Halton points. We provide the exact values of Fig. 4 in the Supporting Information (SI) such that it can serve as a benchmark for future methods. We further provide the $\mathcal{L}_2$-discrepancy values of MPMC together with a discussion outlining their relevance in the SI.

3.B. Optimality of MPMC point sets. Much of the past research on low-discrepancy point sets focused on achieving star-discrepancy with optimal asymptotic order in N for implementation in quasi-Monte Carlo methods. However, there has been a recent surge in interest in finding point sets that minimize discrepancy for fixed N and d. The main contribution in this direction was given in (29), where the authors constructed optimal star-discrepancy point sets in two and three dimensions. Naturally, we are interested in comparing MPMC points to these optimal formulations. The results of the optimal star-discrepancy comparison in two dimensions are presented in Table 1 and the three dimensional case is found in the SI.

Table 1. Comparison in two dimensions of MPMC star-discrepancy values against optimal sets and Fibonacci sets.

N	1	2	3	4	5	6	7
Fibonacci	1.0	0.6909	0.5880	0.4910	0.3528	0.3183	0.2728
Optimal	$1/\varphi$	0.3660	0.2847	0.2500	0.2000	0.1667	0.1500
MPMC	$1/\varphi$	0.3660	0.2847	0.2500	0.2000	0.1692	0.1508
N	8	9	10	11	12	13	14
Fibonacci	0.2553	0.2270	0.2042	0.1857	0.1702	0.1571	0.1459
Optimal	0.1328	0.1235	0.1111	0.1030	0.0952	0.0889	0.0837
MPMC	0.1354	0.1240	0.1124	0.1058	0.0975	0.0908	0.0853
N	15	16	17	18	19	20	21
Fibonacci	0.1390	0.1486	0.1398	0.1320	0.1251	0.1188	0.1131
Optimal	0.0782	0.0739	0.0699	0.0666	0.0634	0.0604	0.0580
MPMC	0.0794	0.0768	0.0731	0.0699	0.0668	0.0640	0.0606

We can see that the star-discrepancy of MPMC points is very close to the star-discrepancy of the optimal point sets, and in fact match it exactly for small N. Moreover, the star-discrepancy of the Fibonacci set is far off the optimal values, i.e., approximately by a factor of 2. Finally, it is worth highlighting, as reported in (29), that the computation of the optimal points requires to solve a non-linear programming problem and takes approximately 18 days to compute for the case of $d = 3$ and $N = 8$. In contrast to that, MPMC was trained from scratch in 72 seconds on an NVIDIA GeForce RTX 2080 Ti GPU for the same case.

3.C. MPMC generation in high dimensions. As discussed in Section 2.D, the efficacy of discrepancy measures to justly evaluate irregularity of distribution is flawed in higher dimensions. Therefore, to alternatively assess the quality of the distribution of higher dimensional point sets and sequences, motivated by the Koksma-Hlawka inequality Eq. (2), we will implement high dimensional MPMC points in an integral arising in a real-world problem from computational finance previously studied in (52, 57, 58).

The primary goal is to accurately estimate the value at time 0 of an Asian call option on an underlying asset that follows a log-normal distribution. Complete details of the problem formulation are provided in the SI. With our chosen

parameters, this problem is known (52) to exhibit more than 97% of its variability in dimensions one, two, and three. Table 2 shows the absolute errors observed when implementing Hammersley, lattice, Sobol' and MPMC constructions. We train a 32-dimensional MPMC point set while emphasizing the 1-3-dimensional projections as described in Section 2.D. Likewise, we utilize the custom QMC software LatNet Builder (59) to construct a rank-1 lattice by the component-by-component construction (60, 61) placing importance on the 1-3-dimensional projections. We report the average absolute error of an MPMC training batch, which is selected based on the minimal Hickernell $\mathcal{L}_2$-discrepancy restricted to 1-3-dimensional projections.

Table 2. Approximation error of an Asian call option pricing of MPMC, Hammersley, a rank-1 lattice, and Sobol'.

N	32	64	128	256	512	1024
Hammersley	6.449	4.125	3.575	2.817	1.947	1.296
Rank-1 lattice	5.636	4.638	1.331	2.151	0.180	0.203
Sobol'	1.235	1.373	0.965	0.623	0.497	0.290
MPMC	1.402	0.831	0.512	0.250	0.120	0.055

The 32-dimensional MPMC point sets show significant enhancements over previous methods. This improvement is particularly notable at higher values of N, where MPMC outperforms rank-1 lattice by a factor of approximately 4, Sobol' by a factor of 5, and Hammersley by a factor of 24. A particularly promising feature is the gain shown by MPMC points through targeted training, surpassing LatNet Builder's targeted rank-1 lattice. The efficiency observed in this high-dimensional problem suggests a superior uniformity of MPMC points. In fact, the observed improvements may partially be explained by the uniformity held in lower-dimensional projections when compared to the traditional choices of QMC point sets. We refer to the SI for further discussion.

3.D. Ablations. Our proposed MPMC method is the result of several design choices, such as the type of input points, the deep learning model, and the training objective. In order to further justify our choices, we ablate several aspects of our MPMC framework illustrated by the following questions:

How does the graph structure influence the performance? To answer this, we compute the average $\mathcal{L}_2$-discrepancy of several trained MPMC models for increasing values of the nearest neighbor radius r in Eq. (4) ranging from 0 to $\sqrt{d}$ for different number of points $N = 64, 128, 1024$, and plot the results in the SI. These results lead to two important observations. First, not using a graph structure at all, i.e., setting $r = 0$ resulting in Deepsets (62), significantly impairs the performance of MPMC, reaching average $\mathcal{L}_2$-discrepancy values that are 9 to 40 times worse than using a graph structure. The second observation is that although the performance of MPMC is relatively stable for any choice of $r > 0$, including the radius r in hyperparameter tuning can help achieve point sets with minimal discrepancy.

What is the role of the GNN architecture used in MPMC? While we base MPMC on message-passing neural networks (MPNNs) (42), other GNN architectures such as Graph Convolutional Networks (GCNs) (38), or Graph Attention Networks (GATs) can be used in this context as well. To check this, we train MPMC based on MPNNs, GCNs, and GATs for three different number of points $N = 64, 256, 1024$ and show the $\mathcal{L}_2$-discrepancy in the SI. Based on these results, we conclude that GCNs and GATs outperform each other based on the number of points N. At the same time, MPNNs consistently yield point sets with the lowest discrepancy values among all three considered GNN architectures.

Does the choice of input point sets described in Section 2.A influence the performance of MPMC? To answer this, we train several MPMC models on all three different types of input points, i.e., random points, Sobol', and a randomized Sobol', where we choose $\xi \sim \mathcal{U}([0, 0.1]^d)$ in Eq. (3), for two different number of points $N = 256, 1024$. We report the average $\mathcal{L}_2$-discrepancy of all trained MPMC models for increasing number of training steps in the SI. We observe that on average Sobol' and randomized Sobol' yield slightly lower discrepancy values and faster convergence compared to random points and thus lead to a more robust performance. However, we further note, that instead of averaging over all trained MPMC models, but instead choosing the single best model yield similar results for each input point type.

4. Discussion

Low-discrepancy points play a central role in many applications in science and engineering. In this article, we have proposed MPMC, the first machine learning approach to generate new sets of low-discrepancy points. Inspired by the geometric nature of constructing such point sets, we base our MPMC approach on GNNs. Choosing an adequate training objective, i.e., closed-form solution of the $\mathcal{L}_2$-discrepancy, we show that MPMC successfully transforms (random) input points into point sets with low discrepancy. Moreover, we extend this framework to higher dimensions, by training with an approximation of the Hickernell $\mathcal{L}_2$-discrepancy. We further present an extensive empirical evaluation to illustrate different aspects of the proposed MPMC approach, highlighting the superior uniformity properties of MPMC points compared to previous state-of-the-art methods. Finally, we carefully ablate key components of our MPMC model, yielding deeper empirical insights.

MPMC represents a novel and efficient way of generating point sets with very low discrepancy. In fact, MPMC is empirically shown to obtain optimal or near-optimal discrepancy for every dimension and the number of points for which the optimal discrepancy can be determined. This is crucial for computationally expensive applications, where MPMC will lead to potentially significantly lower absolute errors compared to previous methods. Moreover, the generality of the MPMC framework allows for designing tailor-made QMC points that exploit specific structures of the problem at hand.

The aim of this paper was to generate point sets with low discrepancy for a fixed dimension and fixed number of points. On the other hand, many important applications require repeated sampling resulting in low-discrepancy sequences and not fixed point sets. Thus, one important aspect of future work will be to extend our MPMC point sets to MPMC sequences. Lastly, based on the superior discrepancy performance, we expect MPMC point sets to excel in various applications. Motivated by this, we would like to apply MPMC to various problems in science and engineering as future work.

ACKNOWLEDGMENTS. The authors would like to thank François Clément (Sorbonne Université, CNRS) for several helpful discussions, and for providing computer code for the further visual and empirical insights contained in the SI. This research was supported in part by the AI2050 program at Schmidt Futures (grant G-22-63172), the Boeing Company, and the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under cooperative agreement number FA8750-19-2-1000. The work of TKR is supported by Postdoc.Mobility grant P500PT-217915 from the Swiss National Science Foundation. The work of NK and CL is supported by the Natural Science and Engineering Research Council of Canada (NSERC) via grant 238959. MB is supported in part by EPSRC Turing AI World-Leading Research Fellowship No. EP/X040062/1.

1. P Glasserman, *Monte Carlo methods in financial engineering*. (Springer, New York), (2004).
2. C Lemieux, *Monte Carlo and quasi-Monte Carlo sampling*, Springer Series in Statistics. (Springer, New York), pp. xvi+373 (2009).
3. M Drmota, RF Tichy, *Sequences, discrepancies and applications*, Lecture Notes in Mathematics. (Springer-Verlag, Berlin) Vol. 1651, pp. xiv+503 (1997).
4. L Kuipers, H Niederreiter, *Uniform distribution of sequences*, Pure and Applied Mathematics. (Wiley-Interscience [John Wiley & Sons], New York-London-Sydney), pp. xiv+390 (1974).
5. E Hlawka, *The theory of uniform distribution*. (A B Academic Publishers, Berkhamsted), (1984).
6. ML Cauwet, et al., Fully parallel hyperparameter search: Reshaped space-filling in *International Conference on Machine Learning*. (PMLR), pp. 1338–1348 (2020).
7. S Galanti, A Jung, Low-discrepancy sequences: Monte Carlo simulation of option prices. *J. Deriv.* pp. 63–83 (1997).
8. L Paulin, et al., Matbuilder: Mastering sampling uniformity over projections. *ACM Transactions on Graph. (TOG)* **41**, 1–13 (2022).
9. S Mishra, TK Rusch, Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. *SIAM J. on Numer. Analysis* **59**, 1811–1834 (2021).
10. M Longo, S Mishra, TK Rusch, C Schwab, Higher-order quasi-Monte Carlo training of deep neural networks. *SIAM J. on Sci. Comput.* **43**, A3938–A3966 (2021).
11. MM Bronstein, J Bruna, T Cohen, P Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv:2104.13478* (2021).
12. P L'Ecuyer, Quasi-Monte Carlo methods with applications in finance. *Finance Stochastics* **13**, 307–349 (2009).
13. MS Branicky, SM LaValle, K Olson, L Yang, Quasi-randomized path planning in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. (IEEE), Vol. 2, pp. 1481–1487 (2001).
14. B Mildenhall, et al., Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* **65**, 99–106 (2021).
15. N Korobov, Number-theoretic methods of approximate analysis. *Fitzmatgiz, Mosc.* (1963) In Russian.
16. S Haber, Numerical evaluation of multiple integrals. *SIAM Rev.* pp. 481–526 (1970).
17. IH Sloan, S Joe, *Lattice Methods for Multiple Integration*. (Oxford University Press), (1994).
18. D Nuyens, *The construction of good lattice rules and polynomial lattice rules*, eds. P Kritzer, H Niederreiter, F Pillichshammer, A Winterhof. (De Gruyter, Berlin, Boston), pp. 223–256 (2014).
19. J Dick, P Kritzer, F Pillichshammer, *Constructions of Lattice Rules*. (Springer International Publishing, Cham), pp. 95–139 (2022).
20. I Sobol', On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Comput. Math. Math. Phys.* **7**, 86–112 (1967).
21. J van der Corput, Verteilungsfunktionen i–ii. *Proc. Akad. Amsterdam* **38**, 813–821, 1058–1066 (1935).
22. H Faure, Discrepance de suites associées à un système de numération (en dimension s). *Acta Arith.* **41**, 337–351 (1982).
23. H Niederreiter, Point sets and sequences with small discrepancy. *Monatshefte für Math.* **104**, 273–337 (1987).
24. H Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*. (1992).
25. JH Halton, On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* **2**, 84–90 (1960).
26. C Doerr, FM De Rainville, Constructing low star discrepancy point sets with genetic algorithms in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. (Association for Computing Machinery, New York, NY, USA), p. 789–796 (2013).
27. F Clément, C Doerr, L Paquete, Star discrepancy subset selection: problem formulation and efficient approaches for low dimensions. *J. Complex.* **70**, Paper No. 101645, 34 (2022).
28. F Clément, C Doerr, L Paquete, Heuristic approaches to obtain low-discrepancy point sets via subset selection. *J. Complex.* **83**, Paper No. 101852 (2024).
29. F Clément, C Doerr, K Klamroth, L Paquete, Constructing optimal $\mathcal{L}_\infty$ star discrepancy sets. *Preprint* (2023) https://arxiv.org/abs/2311.17463.
30. A Sperduti, Encoding labeled graphs by labeling RAAM in *NIPS*. (1994).
31. C Goller, A Kuchler, Learning task-dependent distributed representations by backpropagation through structure in *ICNN*. (1996).
32. A Sperduti, A Starita, Supervised neural networks for the classification of structures. *IEEE Trans. Neural Networks* **8**, 714–735 (1997).
33. P Frasconi, M Gori, A Sperduti, A general framework for adaptive processing of data structures. *IEEE Trans. Neural Networks* **9**, 768–786 (1998).
34. M Gori, G Monfardini, F Scarselli, A new model for learning in graph domains in *IJCNN*. (2005).
35. F Scarselli, M Gori, AC Tsoi, M Hagenbuchner, G Monfardini, The graph neural network model. *IEEE Trans. Neural Networks* **20**, 61–80 (2008).
36. J Bruna, W Zaremba, A Szlam, Y LeCun, Spectral networks and locally connected networks on graphs in *2nd International Conference on Learning Representations, ICLR 2014*. (2014).
37. M Defferrard, X Bresson, P Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering. *Adv. neural information processing systems* **29**, 3844–3852 (2016).
38. TN Kipf, M Welling, Semi-supervised classification with graph convolutional networks in *ICLR*. (2017).
39. F Monti, et al., Geometric deep learning on graphs and manifolds using mixture model cnns in *CVPR*. (2017).
40. A Derrow-Pinion, et al., Eta prediction with graph neural networks in google maps in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. pp. 3767–3776 (2021).
41. R Ying, et al., Graph convolutional neural networks for web-scale recommender systems in *KDD*. (2018).
42. J Gilmer, SS Schoenholz, PF Riley, O Vinyals, GE Dahl, Neural message passing for quantum chemistry in *ICML*. (2017).
43. T Gaudelet, et al., Utilizing graph machine learning within drug discovery and development. *Briefings Bioinforma.* **22** (2021).
44. J Shlomi, P Battaglia, JR Vlimant, Graph neural networks in particle physics. *Mach. Learn. Sci. Technol.* **2**, 021001 (2020).
45. J Zhou, et al., Graph neural networks: a review of methods and applications. *arXiv:1812.08434v4* (2019).
46. A Paszke, et al., Automatic differentiation in pytorch. *Openreview.com* (2017).
47. TT Warnock, Computational investigations of low-discrepancy point sets in *Applications of number theory to numerical analysis*. (Elsevier), pp. 319–343 (1972).
48. WJ Morokoff, RE Caflisch, Quasi-random sequences and their discrepancies. *SIAM J. on Sci. Comput.* **15**, 1251–1279 (1994).
49. X Wang, IH Sloan, Low discrepancy sequences in high dimensions: How well are their projections distributed? *J. Comput. Appl. Math.* **213**, 366–386 (2008).
50. F Hickernell, A generalized discrepancy and quadrature error bound. *Math. computation* **67**, 299–322 (1998).
51. RE Caflisch, WJ Morokoff, AB Owen, Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. *J. Comput. Finance* **1**, 27–46 (1997).
52. C Lemieux, AB Owen, Quasi-regression and the relative importance of the ANOVA components of a function in *Monte Carlo and Quasi-Monte Carlo Methods 2000*, eds. KT Fang, H Niederreiter, FJ Hickernell. (Springer Berlin Heidelberg, Berlin, Heidelberg), pp. 331–344 (2002).
53. DP Dobkin, D Eppstein, DP Mitchell, Computing the discrepancy with applications to supersampling patterns. *ACM Trans. Graph.* **15**, 354–376 (1996).
54. F Clément, et al., Computing star discrepancies with numerical black-box optimization algorithms in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23. (Association for Computing Machinery, New York, NY, USA), p. 1330–1338 (2023).
55. C Doerr, M Gnewuch, M Wahlström, *Calculation of Discrepancy Measures and Applications*, eds. W Chen, A Srivastav, G Travaglini. (Springer International Publishing), pp. 621–678 (2014).
56. N Kirk, C Lemieux, J Wiart, Golden ratio nets and sequences. *Preprint* (2023) http://arxiv.org/abs/2312.11696.
57. X Wang, IH Sloan, Why are high-dimensional finance problems often of low effective dimension? *SIAM J. on Sci. Comput.* **27**, 159–183 (2005).
58. H Faure, C Lemieux, Generalized Halton sequences in 2008: A comparative study. *ACM Trans. Model. Comput. Simul.* **19** (2009).
59. P L'Ecuyer, P Marion, M Godin, F Puchhammer, A tool for custom construction of QMC and RQMC point sets in *International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*. (Springer), pp. 51–70 (2020).
60. FY Kuo, S Joe, Component-by-component construction of good lattice rules with a composite number of points. *J. Complex.* **18**, 943–976 (2002).
61. J Dick, P Kritzer, G Leobacher, F Pillichshammer, A reduced fast component-by-component construction of lattice points for integration in weighted spaces with fast decreasing weights. *J. Comput. Appl. Math.* **276**, 1–15 (2015).
62. M Zaheer, et al., Deep sets. *Adv. neural information processing systems* **30** (2017).
63. B Borda, Optimal and typical $\mathcal{L}_2$-discrepancy of 2-dimensional lattices. *Annali di Matematica* (2024).
64. N Kirk, F Pausinger, On the expected $\mathcal{L}_2$-discrepancy of jittered sampling. *Unif. Distrib. Theory* **18**, 65–82 (2023).
65. R Kritzinger, Uniformly distributed sequences generated by a greedy minimization of the L_2 discrepancy. *Mosc. J. Comb. Number Theory* **11**, 215 – 236 (2022).
66. G Ökten, M Shah, Y Goncharov, Random and deterministic digit permutations of the Halton sequence in *Monte Carlo and quasi-Monte Carlo methods 2010*, Springer Proc. Math. Stat. (Springer, Heidelberg) Vol. 23, pp. 609–622 (2012).
67. X Wang, FJ Hickernell, Randomized Halton sequences. *Math. Comput. Model.* **32**, 887–899 (2000).
68. P L'Ecuyer, C Lemieux, *Recent Advances in Randomized Quasi-Monte Carlo Methods*, eds. M Dror, P L'Ecuyer, F Szidarovszky. (Springer US, New York, NY), pp. 419–474 (2002).
69. AB Owen, Randomly permuted (t, m, s)-nets and (t, s)-sequences in *Niederreiter, H., Shiue, PJS. (eds) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. Lecture Notes in Statistics, vol 106. Springer, New York, NY.* (1995).
70. R Cranley, TNL Patterson, Randomization of number theoretic methods for multiple integration. *SIAM J. on Numer. Analysis* **13**, 904–914 (1976).
71. P Velickovic, et al., Graph attention networks in *6th International Conference on Learning Representations, ICLR*. (2018).

Supporting Information (SI) for
Message-Passing Monte Carlo: Generating low-discrepancy point sets via Graph Neural Networks

A. Training details

All experiments have been run on NVIDIA GeForce RTX 2080 Ti, GeForce RTX 3090, TITAN RTX and Quadro RTX 6000 GPUs. Each model was trained for initial 100k training steps, after which the learning rate was reduced by a factor of 10 whenever the discrepancy measure of the output point sets did not improve for a total of 2k training steps evaluated after every 100 training steps. The training was stopped once the learning rate reached a value less than 10^{-6}. Moreover, the hyperparameters of the model were tuned based on random search according to Table 3, which shows the search-space of each hyperparameter as well as the random distribution used to sample from it.

Table 3. Hyperparameter search-space and random distributions to sample from it.

	range	distribution
learning rate	$[10^{-4}, 10^{-2}]$	log uniform
hidden size $m_0 = m_1 = \cdots = m_L$	$\{32, 64, 128, 256\}$	disc. uniform
number of GNN layers L	$\{1, 2, \ldots, 10\}$	disc. uniform
size of mini-batches	$\{8, 16, 32\}$	disc. uniform
weight decay	$[10^{-8}, 10^{-2}]$	log uniform

B. On the Asian Option Pricing Experiment

We describe the problem of estimating the value at time 0 of an Asian call option on an underlying asset in detail. The results of which are presented in the main text as Table 2.

The main goal is to estimate an expectation of the form,

$$C_0 = \mathbb{E}\left[e^{-rT}\left(\frac{1}{d}\sum_{j=1}^{d} S(u_j) - K\right)^+\right].$$

We let T be the expiration time of the contract, K the strike price, for $Z \sim N(0,1)$ let $S(u) = S(0)e^{\left(r-\frac{\sigma^2}{2}\right)u+\sigma\sqrt{u}Z}$ be the price of the underlying asset at time u, and $0 < u_1 < \ldots < u_d = T$ are d times at which the asset price is observed in order to compute the average used in the option pricing formula. The expectation is taken under the risk-neutral probability measure. Finally, r is the risk-free rate, the notation x^+ means $\max(0, x)$, Φ^{-1} is the inverse CDF of the standard normal distribution and $\Delta_l = u_l - u_{l-1}$. Assuming the stock price follows a geometric Brownian motion with volatility σ, it can be shown that this expectation can be written as follows:

$$C_0 = e^{-rT}\int_{[0,1]^d}\left(\frac{1}{d}\sum_{j=1}^{d} S(0)e^{(r-\frac{\sigma^2}{2})u_j+\sigma\sum_{l=1}^{j}\sqrt{\Delta_l}\Phi^{-1}(x_l)} - K\right)^+ dx_1\ldots dx_d.$$

In our simulations, the true value $C_0 = 7.06574$ was calculated in advance via QMC simulation with 2M Sobol' points with the following set of parameters: $S(0) = 50, T = 1$ year, $r = 0.05, \sigma = 0.3, K = 45$ and $d = 32$.

Further, for each N, the generating vector for the rank-1 lattice is produced from LatNet builder from the command line by the following syntax: `latnetbuilder -t lattice -c ordinary -s N -d 32 -e full-CBC -f CU:P2 -q 2 --weights file:/path/to/weights.txt`. The file weights.txt contains order-dependent weights of 10 on projection orders 1, 2 and 3 and otherwise a default weight of 0.001.

C. On the $\mathcal{L}_2$-discrepancy

The $\mathcal{L}_2$-discrepancy is a well-researched and widely used measure of distribution irregularity, consistently attracting attention from the QMC research community (63–65). In our MPNN framework, we utilize the $\mathcal{L}_2$-discrepancy function as a training objective because of its efficient computation and differentiable formulation to be used in the gradient-based learning. This allows the generation of point sets with very small star discrepancy. As a by-product, we also create point sets with small $\mathcal{L}_2$-discrepancy. Fig. 5 shows the $\mathcal{L}_2$-discrepancy values for increasing number of points $N = 20, \ldots, 1020$ for MPMC, Halton, Sobol', Subset Selection method, Hammersley, and Fibonacci in 2 dimensions. We further present the numerical values of Fig. 5 in Table 4. We can see that MPMC consistently obtains the lowest $\mathcal{L}_2$-discrepancy values for all number of points N. Moreover, MPMC significantly outperforms the previous state-of-the-art method based on Fibonacci point sets. We highlight that the performance gap between MPMC and any other method considered here is even wider for the $\mathcal{L}_2$-discrepancy than for the star-discrepancy D^* in Figure 4 of the main text. This can be explained by the fact that MPMC is trained to minimize the $\mathcal{L}_2$-discrepancy directly, and thus by design optimizes the $\mathcal{L}_2$-discrepancy instead of the star-discrepancy D^*.

D. Further empirical insights

D.1. Post-construction enhancements. Classical quasi-Monte Carlo (QMC) constructions have known limitations, including poor uniformity in low-dimensional projections of high-dimensional point sets (see Figure 7 and 8). To mitigate these issues, significant amount of research has concentrated on incorporating randomness into these deterministic constructions, as discussed in (66–68). This approach also offers the advantage of using repeated uniform sampling – known as randomized quasi-Monte Carlo (RQMC) – for straightforward unbiased error estimation.

One successful method is Owen scrambling in base b (69) which involves sequentially applying uniformly chosen permutations to the base b digits of each point in the set. This method has proven to be very popular and highly effective. Another simpler technique is the random shift modulo 1 (70), where a vector in $[0, 1]^d$ is chosen uniformly at random and added (modulo 1) to each point in the set.

Table 4. $\mathcal{L}_2$-discrepancy values for Halton, Sobol', Subset Selection, Hammersley, Fibonacci, and MPMC for different number of points $N = 20, \ldots, 1020$ in $d = 2$.

N	20	60	100	140	180	220	260	300	340	380	420	460	500
Halton	0.06511	0.01323	0.00751	0.00627	0.00460	0.00434	0.00389	0.00343	0.00292	0.00238	0.00266	0.00215	0.00226
Sobol'	0.03564	0.01660	0.01006	0.00645	0.00551	0.00690	0.00655	0.00435	0.00276	0.00330	0.00267	0.00258	0.00242
Subset Selection	0.02569	0.01541	0.00823	0.00693	0.00530	0.00397	0.00352	0.00410	0.00310	0.00276	0.00266	0.00250	0.00221
Hammersley	0.04796	0.01812	0.01119	0.00833	0.00664	0.00554	0.00469	0.00416	0.00371	0.00335	0.00305	0.00283	0.00262
Fibonacci	0.04324	0.01465	0.00870	0.00657	0.00492	0.00399	0.00344	0.00306	0.00275	0.00249	0.00221	0.00198	0.00182
MPMC	0.02016	0.00756	0.00479	0.00353	0.00284	0.00241	0.00203	0.00179	0.00162	0.00154	0.00135	0.00122	0.00117
N	540	580	620	660	700	740	780	820	860	900	940	980	1020
Halton	0.00182	0.00179	0.00164	0.00171	0.00190	0.00179	0.00139	0.00122	0.00139	0.00126	0.00147	0.00117	0.00107
Sobol'	0.00220	0.00216	0.00157	0.00152	0.00198	0.00164	0.00138	0.00196	0.00155	0.00154	0.00130	0.00125	0.00121
Subset Selection	0.00206	0.00228	0.00205	0.00178	0.00166	0.00167	0.00141	0.00151	0.00153	0.00152	0.00136	0.00112	0.00120
Hammersley	0.00243	0.00224	0.00211	0.00200	0.00190	0.00181	0.00173	0.00165	0.00158	0.00152	0.00146	0.00140	0.00135
Fibonacci	0.00168	0.00155	0.00145	0.00139	0.00132	0.00127	0.00121	0.00115	0.00110	0.00107	0.00103	0.00099	0.00094
MPMC	0.00104	0.00104	0.00098	0.00090	0.00087	0.00088	0.00082	0.00080	0.00074	0.00075	0.00072	0.00072	0.00068

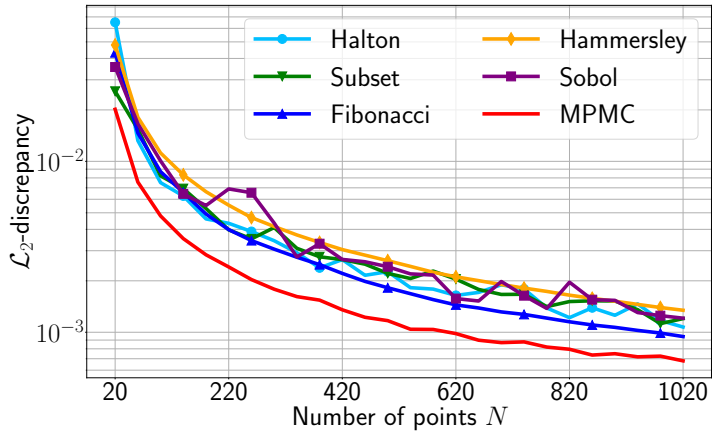

Fig. 5. $\mathcal{L}_2$-discrepancy of Halton, Sobol', Subset Selection, Hammersley, Fibonacci, and MPMC for increasing number of points $N = 20, \ldots, 1020$ in $d = 2$.

In this section, we test how these enhancements improve the results of selected QMC constructions in the Asian option pricing application presented in Section 3.C. of the main text (i.e., Table 2 of the main text). To this end, we start by applying scrambling (in the natural base 2) to the Sobol' sequence and present the mean absolute error (MAE) in Table 5. We can see that on average scrambling massively improves the absolute error of Sobol', i.e., by a factor of more than 10. While Owen scrambling in base b can be applied to an arbitrary point set, the choice of b is paramount to ensure that the low-discrepancy property is preserved after randomization. Since the correct choice of the base is not immediately apparent, scrambling is not directly applicable to MPMC and a direct comparison involving both methods leveraging the scrambling enhancement is not possible. However, uniform random shifting (modulo 1) can be applied to any QMC construction, including MPMC. Therefore, we present the results of randomly shifted MPMC in Table 5. We can see that randomly shifting on average leads to a lower absolute error compared to MPMC without random shifting for smaller number of points N. For large $N = 1024$, however, MPMC with random shifting appears to perform worse than MPMC without random shifting. *This highlights the importance of developing suitable randomization techniques specifically tailored for MPMC, a topic we plan to focus on in future research.*

Table 5. Errors of Sobol', scrambled Sobol', MPMC and randomly shifted MPMC for the Asian option pricing experiment in Section 3.C. of the main text. MPMC and Sobol' errors are provided as absolute errors and taken from Table 2 in the main text, while shifted MPMC and scrambled Sobol' are provided as mean absolute errors (MAE).

N	32	64	128	256	512	1024
Sobol'	1.235	1.373	0.965	0.623	0.497	0.290
Scrambled Sobol'	0.516	0.169	0.076	0.064	0.040	0.022
MPMC	1.402	0.831	0.512	0.250	0.120	0.055
Shifted MPMC	0.521	0.310	0.188	0.128	0.082	0.061

D.2. Structure of MPMC Points. Initially explored in (29), the authors provide insights into the configurations of two dimensional point sets that achieve optimal star-discrepancy by providing visualizations of the local discrepancy within the unit square. Providing equivalent comparisons, Figure 6 displays the local discrepancy plots for Sobol' sequences, optimal point sets and MPMC points. Each plot has its own color scale where darker areas indicate lower local discrepancy values, and brighter areas denote higher values. The presence of a black

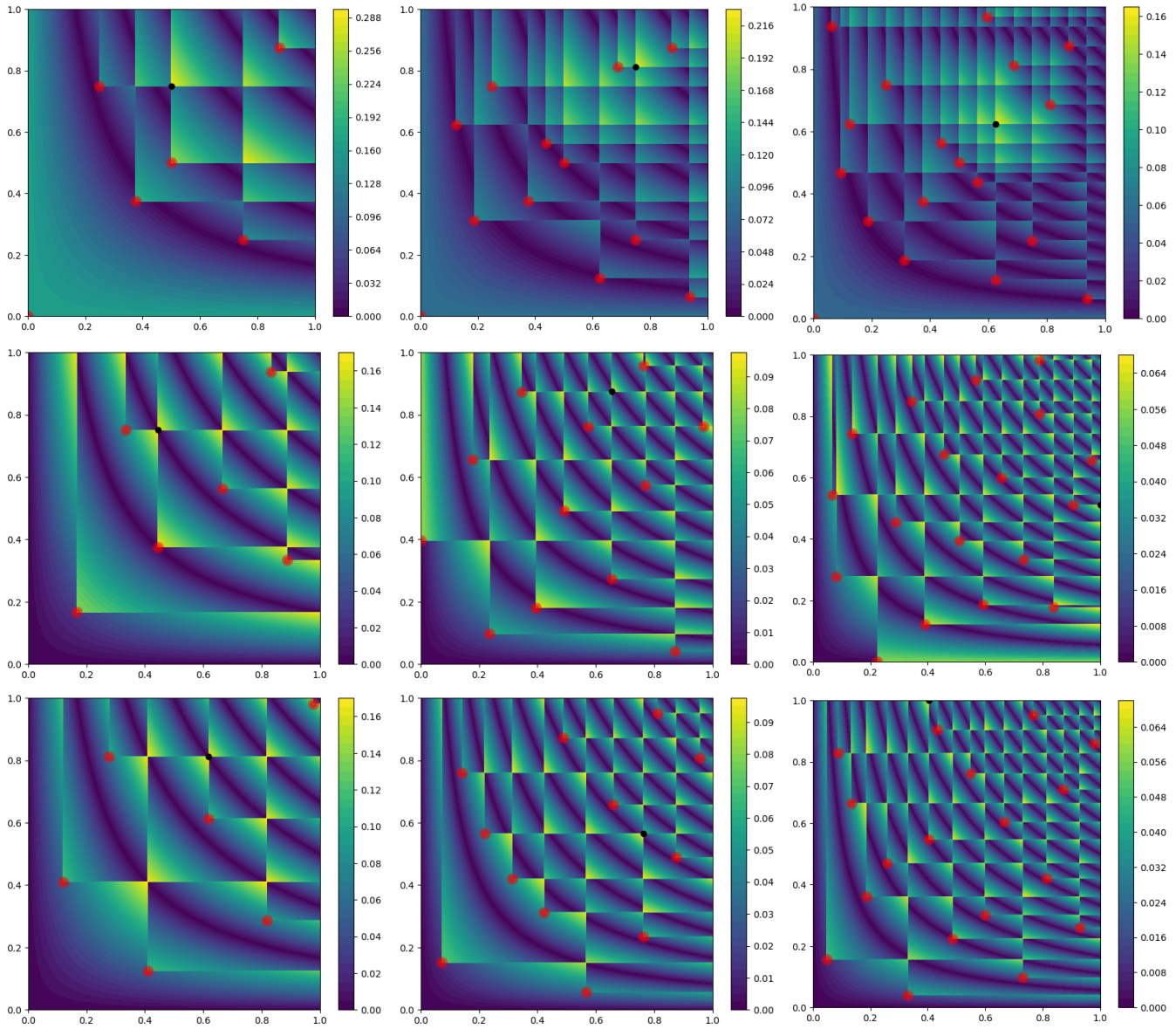

Fig. 6. Local discrepancy plots for Sobol' sequence (**top**), optimal point sets obtained in (29) (**middle row**), and MPMC point sets (**bottom**) for $N = 6$ points (left), $N = 12$ points (middle column), and $N = 18$ points (right).

dot in each plot marks the point of maximum local discrepancy, i.e., the explicit anchored test box where the star-discrepancy is obtained. Additionally, regions of high local discrepancy form bright triangular regions whose corner is either directed toward the upper right corner to represent open boxes with too few points, or angled toward the lower left, indicating closed boxes with an excess of points. For instance, the Sobol' plots exclusively show closed overfilled boxes, with bright triangles pointing downward and leftward toward the origin.

A visual comparison reveals structural similarities between the optimal sets and the MPMC points, suggesting a more balanced distribution of local discrepancy values across the unit square, with both open and closed boxes appearing in the plots. Interestingly, this similar structure emerges despite the sets consisting of quite different exact point values.

In conclusion, it seems evident that the GNN captures an essential underlying local discrepancy structure, which is key for minimizing star-discrepancy.

D.3. Projections of MPMC points. As noted in the main text, when applied to the computational finance integral described in Section B to estimate the value of an Asian call option, the MPMC points far outperform the Sobol' or Hammersley in terms of approximation accuracy. A significantly important factor for the success of QMC methods in high dimensional application is the quality of the distribution in the lower dimensional projections of the QMC point set. See (52) for a more comprehensive discussion. Figure 7 and 8 display the 5-th and 6-th, and 26-th and 27-th coordinate projections respectively of the MPMC points, Sobol', rank-1 lattice and Hammersley constructed in 32 dimensions. Visual inspection reveals that the projections seem to be just as uniformly distributed in the 5-th and 6-th dimensions and notably more evenly distributed as the dimension increases to 26 and 27. At these higher dimensions, we start noticing some undesired correlations in the coordinates of the Sobol' and Hammersley constructions, however, fortunately the MPMC construction does not exhibit this problematic feature displaying no significant correlation, clustering, or sparsity; the MPMC point sets appear random yet maintain a high degree of uniformity. This characteristic is particularly advantageous for tackling high-dimensional problems.

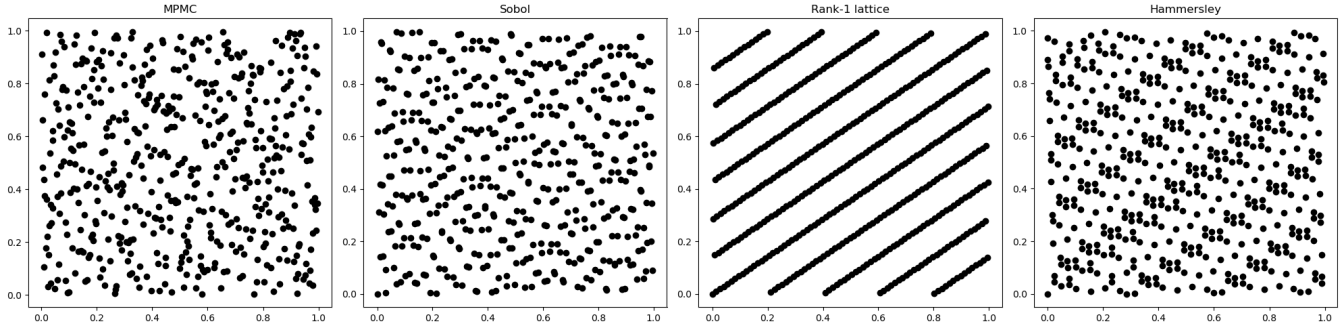

Fig. 7. Projections of the 5-th and 6-th coordinates of 32-dimensional MPMC, Sobol', rank-1 lattice and Hammersley with $N = 512$ (left to right).

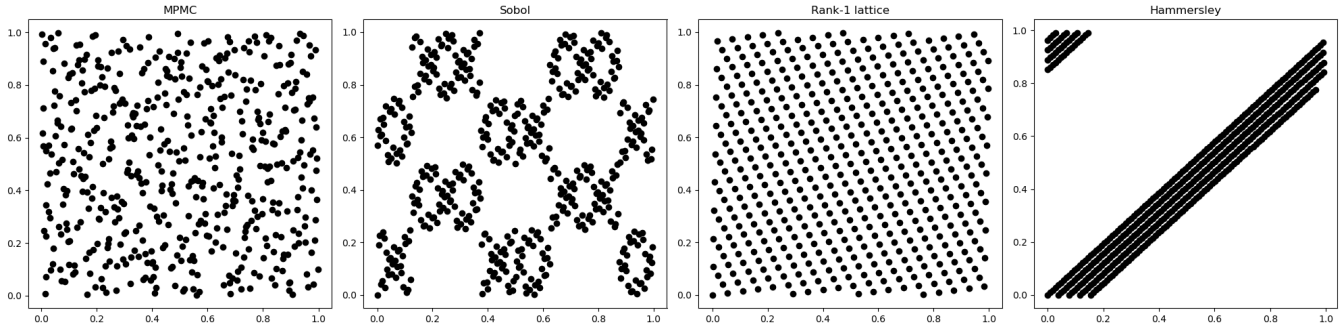

Fig. 8. Projections of the 26-th and 27-th coordinates of 32-dimensional MPMC, Sobol', rank-1 lattice and Hammersley with $N = 512$ (left to right).

D.4. Optimality of MPMC in Three Dimensions. Table 7 shows the star-discrepancy of MPMC in three dimensions for $N = 1, 2, \ldots, 8$ number of points, as well as the optimal star-discrepancy values obtained from (29). We can see that MPMC obtains again near-optimal star-discrepancy for all choices of N.

Table 6. Comparison in three dimensions of MPMC points star-discrepancy values against optimal sets.

N	1	2	3	4	5	6	7	8
Optimal	0.6823	0.4239	0.3445	0.3038	0.2618	0.2326	0.2090	0.1875
MPMC	0.6833	0.4239	0.3491	0.3071	0.2669	0.2371	0.2158	0.1993

D.5. Star-discrepancy values of Figure 4 in the main text. From the main text, we present the exact numerical values of the star-discrepancy for Halton, Sobol', Subset Selection, Hammersley, lifted Sobol', Fibonacci, and MPMC points as a benchmark for future methods.

Table 7. Star-discrepancy values of Figure 4 in the main text for Halton, Sobol', Subset Selection, Hammersley, lifted Sobol', Fibonacci, and MPMC.

N	20	60	100	140	180	220	260	300	340	380	420	460	500
Halton	0.17384	0.06535	0.05024	0.03686	0.03200	0.02323	0.02062	0.01994	0.01950	0.01659	0.01617	0.01279	0.01172
Sobol'	0.13125	0.06583	0.04617	0.03306	0.02466	0.02420	0.02571	0.01851	0.01280	0.01442	0.01326	0.01225	0.01289
Subset Selection	0.08799	0.05469	0.02860	0.02439	0.02028	0.01499	0.01339	0.01527	0.01175	0.01089	0.01073	0.00950	0.00898
Hammersley	0.12304	0.04941	0.03136	0.02292	0.01842	0.01512	0.01308	0.01164	0.01056	0.00945	0.00855	0.00803	0.00739
Lifted Sobol'	0.11875	0.04609	0.02766	0.02388	0.01501	0.01584	0.01221	0.01294	0.00837	0.00994	0.00915	0.00849	0.00721
Fibonacci	0.11885	0.04422	0.02749	0.02128	0.01655	0.01354	0.01200	0.01054	0.00957	0.00857	0.00775	0.00708	0.00651
MPMC	0.06664	0.02729	0.01879	0.01373	0.01147	0.00975	0.00843	0.00752	0.00710	0.00695	0.00584	0.00540	0.00518
N	540	580	620	660	700	740	780	820	860	900	940	980	1020
Halton	0.01101	0.01005	0.00957	0.00949	0.00841	0.00709	0.00685	0.00718	0.00710	0.00571	0.00626	0.00619	0.00636
Sobol'	0.00967	0.00967	0.00977	0.01064	0.00724	0.00765	0.00697	0.00691	0.00772	0.00638	0.00637	0.00640	0.00616
Subset Selection	0.00800	0.00927	0.00802	0.00727	0.00680	0.00664	0.00582	0.00626	0.00626	0.00602	0.00536	0.00457	0.00488
Hammersley	0.00685	0.00637	0.00596	0.00578	0.00545	0.00523	0.00496	0.00472	0.00450	0.00431	0.00413	0.00396	0.00380
Lifted Sobol'	0.00642	0.00645	0.00534	0.00759	0.00509	0.00466	0.00547	0.00441	0.00422	0.00450	0.00373	0.00471	0.00362
Fibonacci	0.00603	0.00561	0.00525	0.00509	0.00480	0.00484	0.00459	0.00436	0.00416	0.00398	0.00381	0.00365	0.00351
MPMC	0.00488	0.00476	0.00454	0.00437	0.00416	0.00397	0.00376	0.00360	0.00356	0.00319	0.00321	0.00308	0.00303

D.6. On the role of the radius in the nearest neighbor graph. We recall from the main text, that our proposed MPMC method is based on GNNs that leverage r-radius nearest neighbors as the underlying computational graph, connecting nodes within a given radius r. How does the performance of MPMC depend on the radius r? Moreover, is it necessary to use GNNs? To answer this, we train 10 MPMC models for different radius values $r = 0.1, \dots, \sqrt{2}$ for different number of points N in $d = 2$ and plot the resulting average $\mathcal{L}_2$-discrepancy in Fig. 9. We can see that there is no correlation between the performance and a fixed radius r. Moreover, the performance appears to be not overly sensitive with respect to different values for the radius. Nevertheless, small variations of the performance with respect to the radius r can be seen and it is thus advisable to include the radius to the set of tune-able hyperparameters of the model.

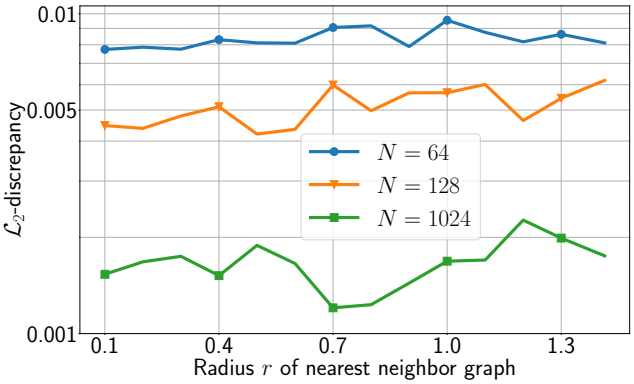

Fig. 9. $\mathcal{L}_2$-discrepancy of MPMC points for increasing values of the radius of the underlying nearest neighbor computational graph ranging from 0.1 to $\sqrt{2}$ for different number of points $N = 64, 128, 1024$ in $d = 2$.

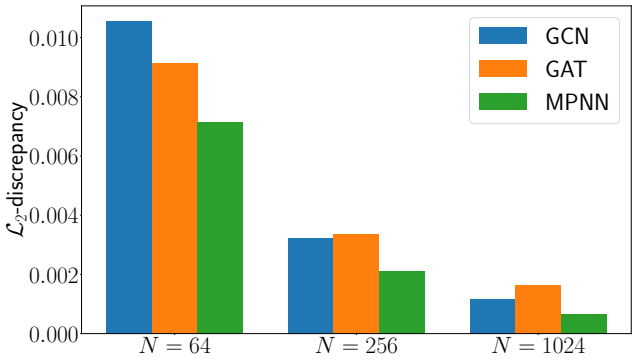

Fig. 10. $\mathcal{L}_2$-discrepancy of MPMC for different choices of GNN architectures, i.e., GCN (38), GAT (71), and MPNN (42) for three different number of points $N = 64, 256, 1024$ in $d = 2$.

We further note that a radius of $r = 0$ corresponds to zero edges in the underlying computational graph. Thus, the model becomes a deepset (62), processing each point in the set individually without aggregating any neighborhood information. The average $\mathcal{L}_2$-discrepancy of this deepset is approximately 0.073 for $N = 64$, 0.058 for $N = 128$, and 0.063 for $N = 1024$, *i.e., between 9 to over 40 times worse than GNNs with* $r \geq 0.1$. Moreover, the deepset fails to decrease the $\mathcal{L}_2$-discrepancy for increasing number of points N. This highlights the necessity of using GNNs that aggregate geometric information from neighboring points for successfully generating low-discrepancy point sets.

D.7. On the role of the GNN architecture. While we base our proposed MPMC model on MPNNs (42), any other GNN architecture could be used instead. Therefore, it is natural to ask how the choice of the GNN architecture influences the performance of MPMC. To answer this, we test three different configurations of MPMC: one based on MPNNs, one based on GCNs (38), and one based on GATs (71). We train all three configurations for different number of points $N = 64, 256, 1024$ in $d = 2$, and provide the results as a bar plot in Fig. 10. We can see that while GCNs and GATs outperform each other depending on the chosen number of points, MPNNs consistently produce point sets with the lowest $\mathcal{L}_2$-discrepancy among all three configurations for all number of points considered here.

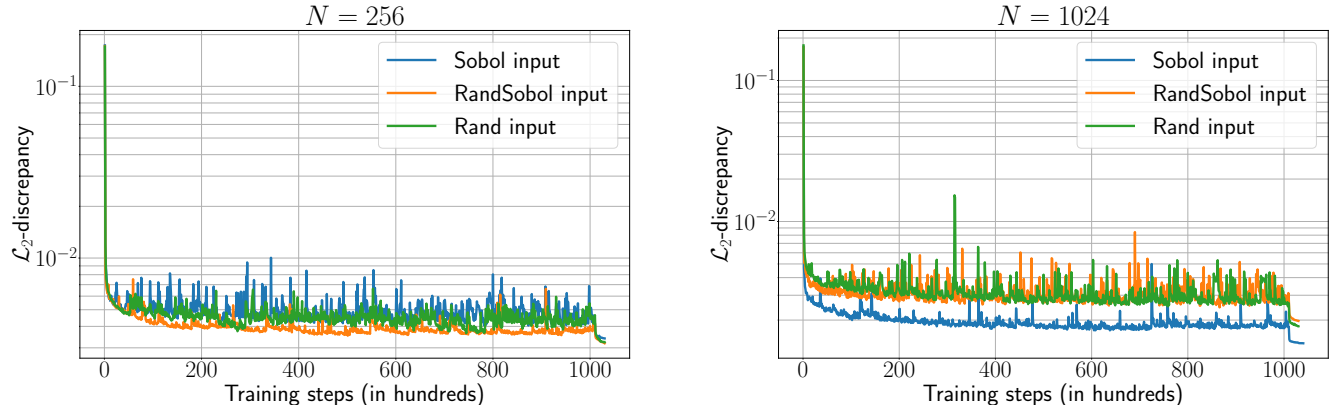

Fig. 11. $\mathcal{L}_2$-discrepancy values of MPMC during training for three different types of input points, i.e., Sobol', Randomized Sobol', and random points, for $N = 256$ and $N = 1024$ in $d = 2$.

D.8. On the role of the input points. In the main text, we suggest three different input types to be transformed into low-discrepancy points via our MPMC framework, namely random points, Sobol', and randomized Sobol'. In this experiment, we empirically analyse how these different types influence the discrepancy of the resulting MPMC points. To this end, we train several MPMC models in $d = 2$ based on the three different input types and report the average $\mathcal{L}_2$-discrepancy during training for two different number of points $N = 256, 1024$ in Fig. 11. We can see that on average either Sobol' or randomized Sobol' reach lower discrepancy values as well as exhibit faster convergence compared to random points. We note, however, that the single best MPMC model for each of the three different input point types yield almost identical discrepancy values. Thus, we conclude that Sobol' and randomized Sobol' points on average yield lower discrepancy values compared to random points, while at the same time the best performing input type has to be evaluated in practice for each number of points N.