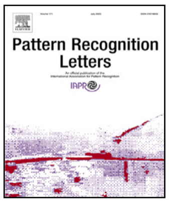
Pattern Recognition Letters 186 (2024) 272–278

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec



Positional diffusion: Graph-based diffusion models for set ordering

Francesco Giuliari [a,c,1], Gianluca Scarpellini [a,c,1], Stefano Fiorini [a,1], Stuart James [d], Pietro Morerio [a,*], Yiming Wang [a,b], Alessio Del Bue [a]

[a] Istituto Italiano di Tecnologia, Via Morego 30, Genoa (GE), Italy
[b] Fondazione Bruno Kessler, Via Sommarive, 18, Povo (TR), Italy
[c] Università degli studi di Genova, Via Balbi, 5, Genova (GE), Italy
[d] Durham University, Upper Mountjoy, Stockton road, Durham, UK

ARTICLE INFO

Editor: Maria De Marsico

Keywords:
Graph neural networks
Diffusion probabilistic models
Positional reasoning

ABSTRACT

Positional reasoning is the process of ordering an unsorted set of parts into a consistent structure. To address this problem, we present *Positional Diffusion*, a plug-and-play graph formulation with Diffusion Probabilistic Models. Using a diffusion process, we add Gaussian noise to the set elements' position and map them to a random position in a continuous space. *Positional Diffusion* learns to reverse the noising process and recover the original positions through an Attention-based Graph Neural Network. To evaluate our method, we conduct extensive experiments on three different tasks and seven datasets, comparing our approach against the state-of-the-art methods for visual puzzle-solving, sentence ordering, and room arrangement, demonstrating that our method outperforms long-lasting research on puzzle solving with up to +17% compared to the second-best deep learning method, and performs on par against the state-of-the-art methods on sentence ordering and room rearrangement. Our work highlights the suitability of diffusion models for ordering problems and proposes a novel formulation and method for solving various ordering tasks. We release our code at https://github.com/IIT-PAVIS/Positional_Diffusion.

1. Introduction

The ability to arrange elements is a fundamental human skill that is learned during the early stages of development and is essential for carrying out daily tasks. Such ability generalizes across different tasks. Researchers suggest that childhood games, such as Jigsaw puzzles, LEGO© blocks, and crosswords play a critical role in building the foundations of reasoning over the correct arrangement of things [1]. While each of these games is tackling a very specific problem, humans have remarkable skills in *"putting an element in the correct place"* regardless of the dimensionality and the information modality of the problems, such as 1-dimensional ($1D$) for arranging texts or $2D$ for solving puzzles. We refer to this ability as *positional reasoning* and formulate it as an *ordering* problem, i.e., assigning a correct position to each element of an unordered set. Positional reasoning involves numerous real-world applications, e.g., art restoration. Ancient frescoes and old texts are usually fragmented and may have missing parts [2]. Computation solutions help archaeologists tackle the problem with classical positional reasoning methods [3,4].

The difficulty in positional reasoning lies in the combinatorial nature of ordering a set of elements into a coherent (given) structure. A robust ordering method must be invariant to the input sets' random permutations. Previous solutions have been designed to be problem-specific. For example, methods addressing Jigsaw puzzle operate on a $2D$ grid by jointly optimizing similarities and permutations [5] or by learning first an image representation complaint with the set of image tiles and then solving a standard Hungarian approach for matching the pieces [6].

Alternatively, sentence ordering is a $1D$ ordering NLP problem where a set of sentences must be ordered by exploiting pairwise similarities and attention mechanisms [7,8] to form a coherent paragraph. Although all these problems involve finding a correct ordering of a set, current solutions are customized to the data modality and contextual information.

We propose *Positional Diffusion*, a unified model for positional reasoning that eliminates the need for architecture re-design when dealing with different input modalities or various dimensionalities of the positioning problem (Fig. 1). In this approach, we represent the elements in

* Corresponding author.
E-mail addresses: francesco.giuliari@iit.it (F. Giuliari), gianluca.scarpellini@iit.it (G. Scarpellini), stefano.fiorini@iit.it (S. Fiorini), stuart.a.james@durham.ac.uk (S. James), pietro.morerio@iit.it (P. Morerio), ywang@fbk.eu (Y. Wang), alessio.delbue@iit.it (A. Del Bue).
[1] Equal contribution.

https://doi.org/10.1016/j.patrec.2024.10.010
Received 10 January 2024; Received in revised form 22 May 2024; Accepted 17 October 2024
Available online 23 October 2024

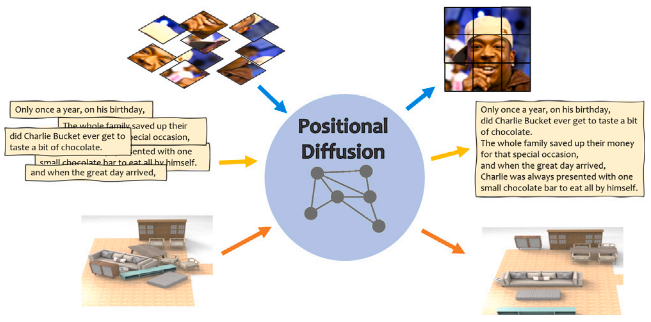

Positional Diffusion
Only once a year, on his birthday,
The whole family saved up their
did Charlie Bucket ever get to taste a bit of chocolate.
special occasion,
presented with one
small chocolate bar to eat all by himself.
and when the great day arrived,
Only once a year, on his birthday, did Charlie Bucket ever get to taste a bit of chocolate.
The whole family saved up their money for that special occasion,
and when the great day arrived,
Charlie was always presented with one small chocolate bar to eat all by himself.

Fig. 1. *Positional Diffusion* is a unified architecture based on Diffusion Probabilistic Models following a graph formulation. It can solve several ordering problems with different dimensionality and multimodal data.

the set as nodes of a complete graph to achieve permutation invariance. We solve this problem by regressing the position of each element in a bounded continuous space. Our approach is based on Diffusion Probabilistic Models (DPMs) to estimate each element's position (and thus ordering) in the set. Using a diffusion formulation during training, we inject noise into the node positions. To learn the reverse process that recovers the correct positions, we employ Graph Neural Networks (GNNs) [9,10], and we train an Attention-based GNN. The relationships among the nodes in the graph are unknown, as we do not assume any prior knowledge about elements' neighborhood. Thus, each node of the graph is connected to all others and the attention mechanism assigns an importance weight to each neighbor before the aggregation phase. At inference, we initialize the graph with sampled positions and iteratively retrieve the correct ordinal positions by conditioning on nodes' features.

In this paper, we demonstrate the effectiveness of our formulation and method with three fundamental tasks: *(i) puzzle solving*, a $2D$ positioning task with visual inputs, where we compare Positional Diffusion to both optimization and learning-based methods, scoring the new State-Of-The-Art (SOTA) performance among all methods with a margin up to +18% compared to the second-best learning-based method; *(ii) sentence ordering*, a $1D$ positioning task with textual inputs, where we obtain performance comparable to the SOTA

without the need of re-training a Large Language Model; and *(iii) Room Rearrangement*, a $2D$ positioning task with abstract object representations as inputs, where we show the benefits of Positional Diffusion formulation over an iterative denoising strategy on 3D Front [11]

Main Contributions and Novelty of the Work:

- We incorporate a graph formulation with DPMs to address the positional reasoning problem. The graph formulation addresses the invariance to input set size and permutations, while the DPMs learn to restore the positions via the noising and de-noising processes;
- We propose a task-agnostic method, *Positional Diffusion*, that implements an Attention-based GNN following a DPM formulation to address positional reasoning in various tasks in a plug-and-play manner;
- We show that, without task-specific customization, *Positional Diffusion* can generalize and achieve SOTA or on-par performance among existing task-specific methods.

2. Related works

We consider related works on recent developments of Diffusion Probabilistic Models and the SOTA methods of the three representative tasks for positional reasoning: *puzzle solving*, *sentence ordering*, and *room rearrangement*.

Diffusion probabilistic models. Diffusion Probabilistic Models (DPMs) solve the inverse problem of removing noise from a noisy data distribution [12]. They gained popularity thanks to their impressive results on image synthesis and their elegant probabilistic interpretation [12]. We propose a formulation of the forward and reverse diffusion process for coherently sorting a shuffled input by treating the problem as either n-dimensional vectors sampled from a Gaussian distribution.

Positional reasoning tasks. Literature on positional reasoning is vast and assumes different connotations depending on the task and modalities involved. Our study focuses on positional reasoning as an ordering task, i.e., sorting shuffled elements into a coherent output.

(i) *Jigsaw Puzzles* [13] interested the optimization community with puzzles as a benchmark for studying image ordering with intrinsic combinatorial complexity [14]. The most successful strategies are related to greedy approaches using hand-crafted features [15,16] with robustness to noise and missing pieces [17] and solving thousands of pieces. This task has also been used as auxiliary task for improving visual encoders [18]

(ii) *Sentence ordering* involves positional reasoning on textual contents, which aims to order sentences into a coherent narration. Several proposed approaches utilize attention-based pointer networks [19], topological sorting [20], deep relational modules [21], and constraint graphs to enhance sentence representations [22]. Other works also re-framed the problem as a ranking problem [23], while [24] formulated sentence ordering as a conditional text generation task using a sequence-to-sequence model [25].

(iii) *Room Rearrangement* is the task of ordering objects in a scene into an *organized* arrangement that is compliant with common sense. Robotics and embodied AI communities recently showed interest in the problem of re-arranging objects in a room to a specific goal [26]. Wei et al. [27] narrowed the problem by focusing on predicting coherent locations for objects in a messy room without any input goal.

Differently from previous literature in computer vision, natural language processing, and multimodal learning, we interpret data shuffling as the noise injection of DPMs' forward process and exploit the reverse process of a DPM to retrieve the final position of each element, that being a sentence, a puzzle piece, an object, or a sentence-image pair. To the best of our knowledge, our *Positional Diffusion* is the first DPM-based solution for positional reasoning that can work with different data modalities and positioning dimensions.

3. Positional diffusion

We define positional reasoning as a restoring process that re-establishes order from shuffled unstructured data distribution in a Euclidean space $\mathbb{R}^n$, where $n = 1$ for $1D$ problems such as sentence ordering, $n = 2$ for $2D$ tasks like puzzle pieces arrangement. Given an unordered set of K elements with some task-specific features $\mathbf{H} = \{\mathbf{h}^1, \dots, \mathbf{h}^K\}, \mathbf{h}^i \in \mathbb{R}^d$, where d is the dimension of the features, and with ground-truth positions $\mathbf{X} = \{\mathbf{x}^1, \dots, \mathbf{x}^K\}, \mathbf{x}^i \in \mathbb{R}^n$, our network estimates a set of positions $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}^1, \dots, \hat{\mathbf{x}}^K\}, \hat{\mathbf{x}}^i \in \mathbb{R}^n$, that matches the real position of each element. Because we use an iterative diffusion process defined by T steps, we use subscript t, to refer to inputs/outputs or features used in the t step of the diffusion chain.

In our method, we adopt a graph-based approach that enables interactions among different elements and accommodates a variable number of input samples. In particular, we define a complete graph G_N with N vertices and $\frac{N(N-1)}{2}$ edges [28], where each data point is represented as a node with an associated feature vector $z_t^i = [\mathbf{x}_t^i; \mathbf{h}^i]^\top$. As shown in Fig. 2, *Positional Diffusion* uses the DPMs formulation to iteratively restore the position of the unordered data from a randomly sampled position and use GNNs to work with our graph-structured data.

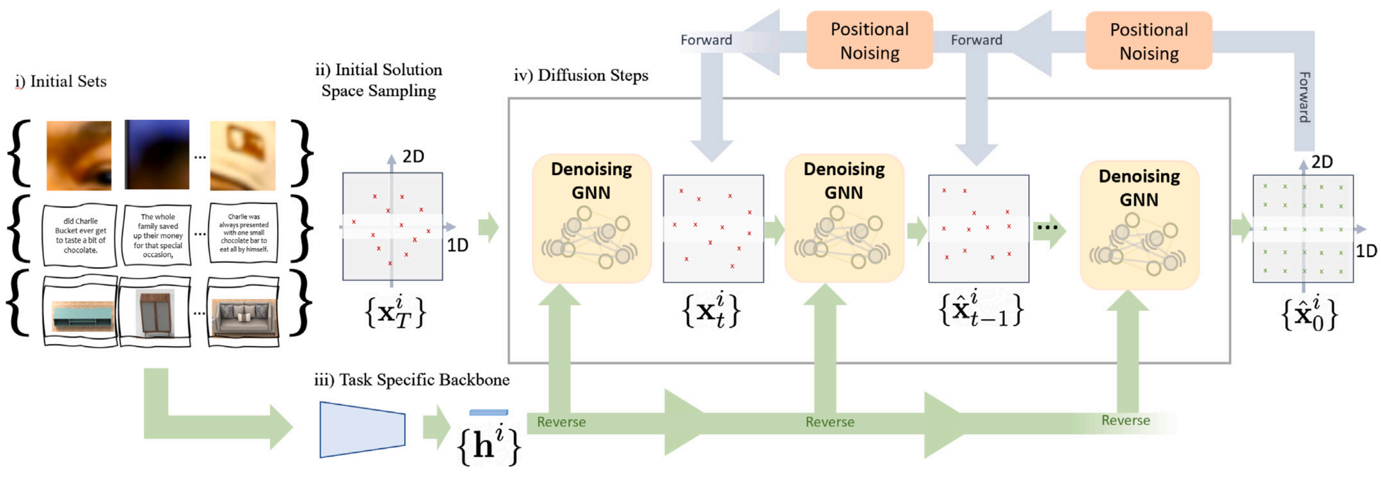

i) Initial Sets
did Charlie Bucket ever get to taste a bit of chocolate.
The whole family saved up their money for that special occasion,
Charlie was always presented with one small chocolate bar to eat all by himself.
ii) Initial Solution Space Sampling
2D
1D
$\{\mathbf{x}_T^i\}$
iii) Task Specific Backbone
$\{\mathbf{h}^i\}$
iv) Diffusion Steps
Denoising GNN
$\{\mathbf{x}_t^i\}$
$\{\hat{\mathbf{x}}_{t-1}^i\}$
$\{\hat{\mathbf{x}}_0^i\}$
Positional Noising
Forward
Reverse

Fig. 2. For each task, the input initial set (i) is a permuted version of the solution. Each element of the set is correlated with an initial sample location (ii) $\mathbf{x}_T^i$ (in $1D$ or $2D$) and an encoding $\mathbf{h}^i$ from a task-specific backbone (iii). During the training of the diffusion steps (iv) we apply a noising process to each element position $\mathbf{x}^i$ to obtain a noisy position $\mathbf{x}_t^i$. We concatenate $\mathbf{h}^i$ with the noisy positions x_t^i to create the features and encode them as node features in a fully connected graph. We use a GNN with an attention mechanism to generate the less noisy positions $\mathbf{x}_{t-1}^i$. During inference, for each element, we sample an initial position $\mathbf{x}_T^i$ from $\mathcal{N}(0,1)$ or set it to $\mathbf{0}$, and use *Positional Diffusion* for the full reverse process to obtain the estimated positions $\hat{\mathbf{x}}_0^i$.

Table 1
Results for puzzle solving on *PuzzleCelebA* and *PuzzleWikiArts*. PO-LA was not assessed for larger sizes due to the computational limitations of its memory footprint. †Trained on individual puzzle sizes.

Dataset	PuzzleCelebA				PuzzleWikiArts			
	6 × 6	8 × 8	10 × 10	12 × 12	6 × 6	8 × 8	10 × 10	12 × 12
Paikin and Tal [17]	99.12	**98.67**	98.39	96.51	98.03	<u>97.35</u>	<u>95.31</u>	<u>90.52</u>
Pomeranz et al. [16]	84.59	79.43	74.80	66.43	79.23	72.64	67.70	62.13
Gallagher [15]	98.55	97.04	95.49	93.13	88.77	82.28	77.17	73.40
PO-LA [5]†	71.96	50.12	38.05	–	12.19	5.77	3.28	–
Ganzzle [6]	72.18	53.26	32.84	12.94	13.48	6.93	4.10	2.58
Positional Diffusion w/o Diffusion Process	99.60	95.20	98.62	96.55	<u>98.52</u>	95.30	88.76	75.84
Positional Diffusion - $\mathcal{N}(0,1)$ sampling	<u>99.72</u>	96.78	<u>99.28</u>	<u>98.55</u>	<u>98.52</u>	97.15	94.34	90.26
Positional Diffusion - Zero-centered initialization	**99.77**	<u>97.53</u>	**99.37**	**98.88**	**99.12**	**98.27**	**96.28**	**93.26**

3.1. Network architecture

To solve the reverse process, we train a GNN that given noisy positions $\mathbf{X}_t$, features $\mathbf{H}$ and a time step t, it outputs the noise ϵ_t that is used to calculate $\mathbf{X}_{t-1}$. Our network operates with element features $\mathbf{h}^i$ that can be extracted from any pre-trained task-specific backbone. We apply the Unified Message Passing Model (UniMP), a GNN architecture introduced by [29], to process G_N. UniMP adopts a Multi-Headed Attention mechanism to adaptively learn and control the amount of information that is gathered from neighboring nodes. Multi-head attention is well-suited for graph contexts where we lack prior knowledge of node relationships, i.e., we cannot define an adjacency matrix A.

3.2. Forward and reverse process

Building upon [12], we define the forward process as a fixed Markov chain that adds Gaussian noise to each input's starting position $\mathbf{x}_0^i = \mathbf{x}^i$ according to a Gaussian distribution. At timestep $t \in [0,T]$, we adopt the variance β_t according to a linear scheduler and define $q(\mathbf{x}_t|\mathbf{x}_0)$ as:

$$q(\mathbf{x}_t^i|\mathbf{x}_0^i) = \mathcal{N}(\mathbf{x}_t^i; \sqrt{\overline{\alpha}_t}\mathbf{x}_0^i, (1-\overline{\alpha}_t)\mathbf{I}), \tag{1}$$

where $\alpha_t = 1-\beta_t$, $\overline{\alpha}_t = \prod_{s=1}^{t}\alpha_s$. Using this formulation, we can obtain a noisy position $\mathbf{x}_t^i$ from $\mathbf{x}_0^i$. The reverse process retrieves the correct position for each data point using the noisy positions $\mathbf{x}_t^i$, and element features $\mathbf{h}^i$. We adopt DDIM [30] algorithm and sample $\hat{\mathbf{x}}_{t-1}$ as:

$$\hat{\mathbf{x}}_{t-1} = \sqrt{\overline{\alpha}_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\overline{\alpha}_t}\epsilon_\theta(\mathbf{x}_t,\mathbf{t},\mathbf{h})}{\sqrt{\overline{\alpha}_t}}\right) + \sqrt{1-\overline{\alpha}_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(\mathbf{x}_t,\mathbf{t},\mathbf{h}) + \sigma_t\epsilon,$$

where $\epsilon_\theta(\mathbf{x}_t,\mathbf{t},\mathbf{h})$ is the estimated noise that has to be removed from $\mathbf{x}_t$ to recover $\hat{\mathbf{x}}_{t-1}$. In the formula, we omit the superscripts i as the network operates on all elements simultaneously as a graph. DDIM introduces the parameter σ to control the stochastic sampling. As the ordering tasks have only one correct arrangement, we set $\sigma = 0$ to make the sampling deterministic.

Our method is trained using the loss for diffusion models introduced in [12]:

$$L_{\text{simple}}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\epsilon}[\|\epsilon - \epsilon_\theta(\underbrace{\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\epsilon}_{\mathbf{x}_t},\mathbf{t},\mathbf{h})\|].$$

We calculate $\mathbf{x}_t$ in closed form from $\mathbf{x}_0$, using the reparametrization trick with the noise vector ϵ. The network learns to minimize the Mean Squared Error between ϵ and the output $\hat{\epsilon} = \epsilon_\theta(\mathbf{x}_t,\mathbf{t},\mathbf{h})$.

3.3. Zero-centered initialization

In generative diffusion models, the initial $\mathbf{X}_T$ set during the reverse process is sampled from $\mathcal{N}(0,1)$. This noise is used in standard image generation tasks to introduce stochasticity for creating different images. However, since the solution in positional reasoning represents the final arrangement, it should only be influenced by the input features $\mathbf{H}$ and not by the initial $\mathbf{X}_T$. Moreover, to guarantee a unique solution, it is essential to ensure that all elements have an equal probability of being moved to the correct positions without introducing initial noise that could generate a different scenario from the correct one. In this regard, the mean of the noise distribution is the optimal starting position. We

Table 2
Results for sentence ordering compiled from [24]. † Requires fine-tuning BART. ‡ Requires fine-tuning BERT.

Method	NeurIPS Abstract			Wikipedia movie plots			ROCStories		
	var. length, avg. 6 (max 15)			*var. length, avg. 13.5 (max 20)*			*fixed length, avg. 5 (max 5)*		
	Acc	PMR	τ	Acc	PMR	τ	Acc	PMR	τ
TGCM [33]	59.43	31.44	0.75	–	–	–	–	–	–
RankTxNet‡ [34]	–	24.13	0.75	–	–	–	–	38.02	0.76
B-TSort‡ [20]	61.48	32.59	0.81	–	–	–	–	–	–
BERSON‡ [21]	*73.87*	48.01	0.85	–	–	–	82.86	68.23	0.88
Bart (seq2seq)† [24]	64.35	33.69	0.78	30.01	18.88	0.59	80.42	*63.50*	*0.85*
Re-BART† [24]	**77.41**	**57.03**	**0.89**	42.04	**25.76**	**0.77**	**90.78**	**81.88**	**0.94**
Positional Diffusion w/o Diffusion Process	70.31	36.58	0.84	48.31	22.51	0.61	71.78	44.59	0.80
Positional Diffusion - $\mathcal{N}(0,1)$ sampling	64.90	29.10	0.78	47.40	21.52	0.62	56.00	25.00	0.61
Positional Diffusion - Zero-centered initialization	74.44	45.24	0.85	**50.41**	25.00	0.63	75.12	54.80	0.82
Positional Diffusion - Zero-centered initialization w/ Re-BART backbone	71.12	39.31	0.85	**52.15**	**29.38**	0.69	80.11	59.95	0.86

propose to set $\mathbf{x}_T = \mathbf{0}$ at the beginning of the reverse process, and show the benefits of this solution through extensive experiments and ablations (see Appendix). We use zero-centered initialization throughout the experiments, unless specified otherwise.

4. Experimental evaluation

We evaluate *Positional Diffusion* on three tasks that require positional reasoning on different dimensions and modalities: (i) puzzle solving operates with visual data to order the shuffled image patches into a complete image in $2D$; (ii) sentence ordering operates with textual data that aims to order the shuffled sentences in $1D$ to form a complete and reasonable paragraph; and (iii) room rearrangement operates with objects' bounding-boxes and classes to re-arrange the objects of a messy room into the most likely coherent configuration in a $2D$ floor plan. We report additional details about the adopted datasets in the Appendix. The following sections introduce the detailed experimental setup for each task regarding the evaluation protocols, performance metrics, and comparisons.

Throughout this section, tables report the best results in **boldface** and the second-best underlined.

4.1. Puzzle solving

We follow the experimental setup in Ganzzle [6] and report the results of *Positional Diffusion* in comparison to optimization-based and deep learning-based methods on *PuzzleCelebA*, based on CelebA-HQ [31], and *PuzzleWikiArts*, based on WikiArts [32] (see Appendix for more details). These two datasets feature many images, allowing for method training with deep learning, while other puzzle datasets typically only contain ≤ 100 images. For both datasets, we test with puzzles of 6, 8, 10, and 12 squared size. As the puzzle size increases, the problem becomes more difficult, as the permutations increase and each piece contains less discriminative information.

The Appendix includes an extensive ablation study, qualitative results, and additional experiments with missing pieces and eroded patches.

Evaluation metrics. We evaluate the performance of *Positional Diffusion* using the *Direct Comparison Metric* [13], an accuracy that indicates the number of correctly ordered pieces over the full test set.

Implementation details. As input an image is provided in $n \times n$ patches, resulting a total of $K = n^2$ elements. We divide a $2D$ target space with a range of $(-1,-1)$, $(1,1)$ into a grid of $n \times n$ cells. We use the centers of the cells as ground truth positions $\mathbf{X}$ for the patches. The input data for puzzle solving are the pixel values for each patch, resized to 32×32. We use EfficientNet [35] as the task-specific backbone to extract the patch visual features $\mathbf{h}^i$ and we train the diffusion model with $T = 300$ and sample it with inference ratio $r = 10$. Regarding the details of UniMP, we configure it with 4 stacked graph attention layers, each employing 8 attention heads. We train a single model with all puzzle sizes simultaneously. At inference, we arrange the patches by mapping each estimated patch position $\hat{\mathbf{x}}^i$ to a cell in the grid.

We measure the distance between each patch position and cells' centers and assign each patch to its closest cell, mapping each cell to at most one patch. By using a greedy approach that prioritizes the assignment between cell-patch pairs starting from those with the lowest distance, the most confident prediction will be assigned first, increasing the prediction robustness.

Comparisons. We compared *Positional Diffusion* against a set of SOTA methods for puzzle solving. *Optimization methods* [15–17] are hand-crafted methods for puzzle solving. They involve computing a compatibility score between all pairs of pieces to predict which are neighbors. *PO-LA* [5] uses a neural network to learn a differentiable permutation invariant ordering cost between a set of patches. *Ganzzle* [6] employs a GAN to generate a hallucinated version of the full image from the set of pieces and solves the puzzle as an assignment problem with the Hungarian algorithm. *Positional Diffusion w/o Diffusion Process* shares the same architecture with *Positional Diffusion* but predicts the positions of each piece in one step. We present two variants of *Positional Diffusion*, where one uses the standard DPM random sampling from $\mathcal{N}(0,1)$, and the other uses the proposed zero-centered initialization for sampling.

Table 1 presents the results of all methods for solving puzzles of various sizes with the two datasets. On *PuzzleCelebA*, both the *Positional Diffusion w/o Diffusion Process* and *Positional Diffusion* outperform the previous SOTA methods on almost all puzzle sizes. In particular, *Positional Diffusion* scores the new SOTA performance among learning-based methods on all puzzle sizes, with a significant improvement against the previous best-performing method Ganzzle [6], even outperforming classical optimization approaches.

PuzzleWikiArts contains puzzles that are harder to solve, as they come from paintings with different pictorial styles and subjects, with little common patterns. Nevertheless, *Positional Diffusion* consistently obtains the best performance among all methods, even outperforming the optimization-based methods, which require hand-crafted features and greedy solutions, on all puzzle sizes. Using the same trained model, *Positional Diffusion* with the zero-centered initialization consistently obtains better performance than using the standard DPM random sampling from $\mathcal{N}(0,1)$.

Since the *Direct Comparison Metric* is computed at the patch level, it does not reflect the performance of solving a puzzle as a whole. For example, the *Positional Diffusion w/o Diffusion Process* positioned 75.84% patches correctly on *PuzzleWikiArt* 12×12, but it only solved 6.64% of the puzzles, while *Positional Diffusion* with 93.26% correctly positioned patches solved 69.32% of puzzles.

Computation analysis. We analyze the average computational time for solving 6×6 puzzles across all methods and we report results in Table 3. *Positional Diffusion*'s time requirements are higher than the other deep-learning baselines when performing 300 steps with an

Table 3
Computation time for solving a 6 × 6 puzzle averaged over 24 samples.

	Method	Time (ms)
Optimization based	Paikin and Tal	27.47
	Pomeranz et al.	221.64
	Gallagher	235.19
Learning based	PO-LA	22.38
	Hung-Perm	9.97
	Ganzzle	25.16
	Positional Diffusion	84.38

inference ratio of 10, while being computationally advantageous with respect to optimization approaches. *Positional Diffusion* trades the additional computation time for achieving state-of-the-art performances, outperforming optimization based approaches.

4.2. Sentence ordering

For *Sentence Ordering*, we follow the experimental setup in [24] and report the results of all compared methods on three common textual datasets (dataset statistics are in the Appendix.): *NeurIPS Abstract* is obtained from the abstracts of scientific articles featured at NeurIPS; *Wikipedia Movie Plots* is a collection of plots of popular movies that are available on Wikipedia; *ROCStories* is a collection of 5 sentences stories regarding everyday events.

Evaluation metrics. We quantify the sentence ordering performances with three metrics as in [24]:

- *Accuracy (Acc.)* is the percentage of correctly predicted sentence positions in an input text.
- *Perfect Match Ratio (PMR)* is the percentage of the number of correctly ordered texts over the total number of texts in the test set. Differently from Acc. which is calculated over individual sentences, PMR measures if the full input text is ordered correctly.
- *Kendall's Tau* (τ) measures the correlation between the ground-truth orders of sentences and the predicted ones, defined as: $\tau = 1 - (2(\#\text{Inversions})\binom{K}{2}^{-1})$, where K is the number of sentences in an input text, and #Inversions is the number of discordant pairs.

We report the metrics averaged over the test set. Higher is better.

Implementation details. We divide a text into a variable number K sentences with shuffled orders as the input. To assign the correct positions $\mathbf{x}_0$ to each sentence, we evenly sample K positions over the interval $(-1, 1)$, and assigned them to the divided sentences based on their position in the text. The starting sentence will have the smallest position, while the ending sentence will have the largest position. We use a frozen pre-trained BART [25] language model for our task-specific feature backbone, to which we added a learnable transformer encoder layer at the end. For each sentence, we prepend a $\langle bos \rangle$ token and pass the sentence to BART to obtain the token feature as the task-specific feature $\mathbf{h}^i$ in *Positional Diffusion*. We train our method with $T = 300$ and sample with inference ratio $r = 10$. The architecture configuration remains consistent with that reported for the puzzle solving task.

Comparisons. We conducted a comprehensive evaluation of *Positional Diffusion* against the current best-performing methods BERSON [21], Re-BART [24], and BART for seq2seq generation as proposed in [24], as well as other baselines including B-TSort [20], RankTxNet [34], TGCM [33]. We also provide the results of *Positional Diffusion w/o Diffusion Process*, which shares the same architecture of *Positional Diffusion* but directly predicts the final order in a single step.

We report the results in Table 2.

Wikipedia Movie Plots has the largest average number of sentences in a paragraph, which is more than double compared to that of *NeurIPS Abstract* and *On ROCStories*. *Positional Diffusion* scores the best Accuracy on *Wikipedia Movie Plots*, with an improvement of +8% over the current SOTA method Re-BART, with on par performance in terms of PMR and τ. With *NeurIPS Abstract*, *Positional Diffusion* is the second-best method in Accuracy and τ, while Re-BART remains the best-performing one. It is important to note that we use frozen BART to extract the word embedding for a single sentence, and then use a learnable two-layer transformer to convert the word embedding to a single sentence embedding that we use as a node feature to train our GNN model for positional reasoning. Instead, Re-BART [24] fine-tunes BART with all sentences simultaneously to predict the sequence order. In fact, the trainable parameters of *Positional Diffusion* is 32M for text ordering, which is negligible compared to Re-BART's 425M.

On *ROCStories*, *Positional Diffusion* performs worse than BEARSON and Re-BART. Compared to the well-structured texts in *NeurIPS Abstract* and *Wikipedia Movie Plots*, the logical connection among sentences in *ROCStories* is weak in some cases, which we credit it as the main reason for the poorer performance of *Positional Diffusion*.

We further explore adopting *Positional Diffusion* with Re-BART backbone. Re-BART improves *Positional Diffusion* over ROCStories and Wikipedia Movie Plots across all metrics, while BART backbone still outperforms on NeurIPS Abstract.

Re-BART fine-tunes BART by concatenating the input sentences and computing features for the entire paragraph, that are later fed to a decoder that predicts the correct order. On the other hand, our method builds a graph where each node represents sentence to order. Node features are extracted via a text feature-encoder that is fed with individual sentences, which makes Re-BART less ideal as it requires full paragraph as input.

Finally, we highlight a trend that was already evident for Jigsaw Puzzles: *Positional Diffusion* generally excels on ordering big sets with higher variability, such as sentences for NeurIPS abstract and Wikipedia Movie Plots, or **12 × 12** *Wikiart* puzzles.

4.3. Room rearrangement

We evaluate *Positional Diffusion*on rearranging objects in coherent positions in $2D$ space. We adopt the experimental setup in [27] and report results on professionally-arranged *living rooms* and *bedrooms* from the 3D-Front dataset [11]. Contrarily to jigsaw puzzles, objects in a room may be correctly placed in multiple valid positions (e.g., a chair may be appropriately positioned around any table in the room). Wei et al. initialized the objects' locations slightly off the correct ones at testing, and then consider that only the acceptable configuration closest to the ground-truth is correct.

Evaluation metrics. We quantify the room rearrangement performances with two metrics as in [27]:

- *Distance moved*: distance between starting (noisy) and final (rearranged) positions of the objects in the scene;
- *Earth moving distance to ground-truth (EMD to GT)*: earth moving distance between rearranged objects' location and ground-truth;
- *% scenes with objects within boundaries*: Percentage of denoised scenes with at least 90% of its furniture within the floor plan boundaries;

Implementation details. For each object in scene, we represent its (x, y) position in bird's-eye view and orientation θ as a vector $\mathbf{x} = [x, y, \cos\theta, \sin\theta]$. As input, we pass, as node features, the $\mathbf{x}$ vector, the object's *class* and the *2D bounding-boxes*.

Furthermore, we encode the room layout with PointNet [37] following [27] and include it as an additional node in the input graph. At training, we add Gaussian noise (forward pass) with $T = 1500$ and train *Positional Diffusion* to predict the initial position and orientation of every object. At inference, we follow [27] and initialize objects by adding Gaussian noise $\mathcal{N}(0, 0.1)$ to ground-truth positions, and we reverse the noise with *Positional Diffusion*. We take inspiration from [38] and propose to start the de-noising process from timestep T_{infer} much lower than T to account for the difference between training and test distributions of the object positions.

Table 4
Experiment results on room rearrangement compiled from [27].

	Inference	Living Room			Bedroom		
		Distance Moved ↓	EMD to GT ↓	% within boundaries ↑	Distance Moved ↓	EMD to GT ↓	% within boundaries ↑
ATISS [36]	*failure-correction*	0.1473	0.3378	–	0.2025	0.4673	–
Lego-net [27]	grad. w/ noise	<u>0.091</u>	0.125	54.20	**0.052**	0.086	84.20
	grad. w/o noise	**0.086**	0.117	54.40	<u>0.0492</u>	0.0815	84.40
Positional Diffusion	$T_{infer} = T = 1500$	0.140	<u>0.048</u>	<u>58.93</u>	0.0781	<u>0.0683</u>	<u>85.31</u>
	$T_{infer} = 100$	**0.086**	**0.037**	**59.38**	0.066	**0.055**	**89.01**

Table 5
Overview of the experimental settings. [†]We report the parameters of the trainable Transformer built on top of the frozen BART model (425 M).

Task	Position Dim.	Data Modality	Feature Backbone(s)	Trainable parameters Backbone	GNN
Puzzle solving	$2D$	RGB	EfficientNet [35]	6.8 M	3.2 M
Sentences ordering	$1D$	Text	BART [25]	28.2 M[†]	
Room rearrangement	$2D$	Room layout & Obj. class/bbox	PointNet [37] & Embedding layers	1.22 M	

Comparisons. We compare *Positional Diffusion* with ATISS [36] and Lego-net [27]. ATISS differs from its original implementation and performs *failure-correction* by iteratively re-positioning objects with low probability in a scene.

Lego-net iteratively de-noises the starting object positions (*w/o noise*). A variant of Lego-net (*w/noise*) adds a small noise at every iteration. Wei et al. [27] used fixed-variance Gaussian noise at training while injecting time-dependent noise at inference. This strategy is fundamentally different from diffusion models. Ho et al. [12] shows that time-dependent noise injection at training is an essential characteristic of diffusion and score-based generative models that sets them apart from traditional denoising techniques.

Table 4 shows that *Positional Diffusion* outperforms the baselines on the living room dataset. In particular, our approach greatly reduces the *EMD to GT* by 70% on living room with respect to Lego-net, while maintaining *distance moved* for the object low. On bedrooms, our method outperforms the baselines on *EMD to GT* while maintaining a low average distance for objects. A coherent room should have feasible positions of objects, e.g., inside its boundaries. We compare our approach to the *Lego-net* on the % of scenes with objects within boundaries. The results show that *Positional Diffusion* consistently outperforms *Lego-net* in locating objects within rooms.

We initialize the inference process from a much lower timestep T_{infer} and compare truncated inference at $T_{\text{infer}} = 100$ to full inference at $T = 1500$. Truncating the diffusion process improves living room and bedroom results while reducing the number of inference steps by 90%.

4.4. Experiment details

Hardware. The experiments were conducted on a computer with 2 NVIDIA Tesla V100 16 GB, 380 GB RAM, and 2x Intel(R) Xeon(R) Silver 4210 CPU @ 2.20 GHz Sky Lake CPU.

Model settings. We train *Positional Diffusion* with a learning rate of 10^{-4} and employ Adagrad as the optimization algorithm [39]. We set a maximum of 1000 epochs during our training process, but we stop the training earlier to prevent unnecessary iterations when the loss no longer decreases.

Table 5 shows the different dimensionality, modality, and number of parameters for each of our downstream tasks. It is worth noting that our *Positional Diffusion* shares the same structure across all tasks.

Due to limited computational resources and the high cost associated with the various tasks, we could not rerun our model with multiple seeds.

5. Conclusion

In this work, we proposed *Positional Diffusion*, a graph-based DPM for positional reasoning on unordered sets. *Positional Diffusion* represents the set as a fully connected graph where each element is a node of the graph. By using an Attention-based GNN, we update the node features to estimate the node position. The diffusion formulation allows us to learn the underlying patterns and iteratively refine the element positions. As demonstrated in the experimental section, *Positional Diffusion* is generic and applicable to multiple tasks that require positional reasoning regardless of the data modality and positional dimension. We experimented with three ordering tasks: puzzle solving, sentence ordering, and room rearrangement. *Positional Diffusion* reaches SOTA on puzzle solving–outperforming long-lasting optimization methods while being computationally efficient, outperforms previous methods on room rearrangement, and achieves comparable results on sentence ordering on par with methods specifically designed for the task. Furthermore, *Positional Diffusion* outperforms previous methods when injecting noise in puzzle solving tasks, demonstrating robustness and generalization to unseen scenarios.

Limitations and future work. Our experiments highlighted some of the limitations of *Positional Diffusion*. Adopting the diffusion-based formulation improves performances across all tasks and datasets but introduces a computational overhead due to the iterative diffusion process. Furthermore, *Positional Diffusion* learns a distribution of element positions conditioned on the inputs. While task-specific approaches work well for smaller set of elements (e.g., 6×6 puzzles, ROCStories 5-sentence paragraphs), *Positional Diffusion* shines for more complex distribution, e.g., for 12×12 puzzles and Wikipedia 20-sentence paragraphs. While *Positional Diffusion* is easy to apply, it needs to be trained per task.

In future work, our aim is to strengthen the generalization capability by training a single foundation model to address positional reasoning across multiple modalities and tasks. In addition, the fully-connected graph formulation limits the method to scale in terms of set size. We will explore graph formulations with dynamic connectivity to mitigate this limitation.

CRediT authorship contribution statement

Francesco Giuliari: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Gianluca Scarpellini:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Stefano Fiorini:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation. **Stuart James:** Writing –

review & editing, Writing – original draft, Supervision. **Pietro Morerio:** Writing – review & editing, Supervision, Conceptualization. **Yiming Wang:** Writing – review & editing, Writing – original draft, Supervision. **Alessio Del Bue:** Writing – review & editing, Writing – original draft, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We acknowledge the financial support from PNRR MUR Project PE0000013 - CUP J53C22003010006 "Future Artificial Intelligence Research (hereafter FAIR)", funded by the European Union – NextGenerationEU.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.patrec.2024.10.010.

Data availability

Link to code in appendix.

References

[1] S.C. Levine, K.R. Ratliff, J. Huttenlocher, J. Cannon, Early puzzle play: A predictor of preschoolers' spatial transformation skill, Dev. Psychol. 48 (2) (2012) 530.
[2] P. Dondi, L. Lombardi, A. Setti, DAFNE: A dataset of fresco fragments for digital anastlyolsis, Pattern Recognit. Lett. 138 (2020) 631–637.
[3] N. Derech, A. Tal, I. Shimshoni, Solving archaeological puzzles, Pattern Recognit. 119 (2021) 108065.
[4] T.M. Paixão, R.F. Berriel, M.C. Boeres, A.L. Koerich, C. Badue, A.F. De Souza, T. Oliveira-Santos, Self-supervised deep reconstruction of mixed strip-shredded text documents, Pattern Recognit. 107 (2020) 107535.
[5] Y. Zhang, J. Hare, A. Prügel-Bennett, Learning representations of sets through optimized permutations, in: International Conference on Learning Representations, 2019.
[6] D. Talon, A. Del Bue, S. James, GANzzle: Reframing jigsaw puzzle solving as a retrieval task using a generative mental image, in: 2022 IEEE International Conference on Image Processing, ICIP, 2022, pp. 4083–4087.
[7] J. Gong, X. Chen, X. Qiu, X. Huang, End-to-end neural sentence ordering using pointer network, 2016, arXiv preprint arXiv:1611.04953.
[8] L. Logeswaran, H. Lee, D. Radev, Sentence ordering and coherence modeling using recurrent neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, (1) 2018.
[9] G. Renton, M. Balcilar, P. Héroux, B. Gaüzère, P. Honeine, S. Adam, Symbols detection and classification using graph neural networks, Pattern Recognit. Lett. 152 (2021) 391–397.
[10] A. Panda, D.P. Mukherjee, Compositional zero-shot learning using multi-branch graph convolution and cross-layer knowledge sharing, Pattern Recognit. 145 (2024) 109916.
[11] H. Fu, B. Cai, L. Gao, L.-X. Zhang, J. Wang, C. Li, Q. Zeng, C. Sun, R. Jia, B. Zhao, et al., 3D-front: 3D furnished rooms with layouts and semantics, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10933–10942.
[12] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Adv. Neural Inf. Process. Syst. 33 (2020) 6840–6851.
[13] T.S. Cho, S. Avidan, W.T. Freeman, A probabilistic image jigsaw puzzle solver, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 183–190.
[14] K.J. Batenburg, W.A. Kosters, Solving nonograms by combining relaxations, Pattern Recognit. 42 (8) (2009) 1672–1683.
[15] A.C. Gallagher, Jigsaw puzzles with pieces of unknown orientation, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012.
[16] D. Pomeranz, M. Shemesh, O. Ben-Shahar, A fully automated greedy square jigsaw puzzle solver, in: CVPR 2011, 2011, pp. 9–16.
[17] G. Paikin, A. Tal, Solving multiple square jigsaw puzzles with missing pieces, IEEE CVPR (2015) 4832–4839.
[18] Y. Chen, X. Shen, Y. Liu, Q. Tao, J.A. Suykens, Jigsaw-ViT: Learning jigsaw puzzles in vision transformer, Pattern Recognit. Lett. (ISSN: 0167-8655) 166 (2023) 53–60.
[19] O. Vinyals, M. Fortunato, N. Jaitly, Pointer networks, in: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, 2015, pp. 2692–2700.
[20] S. Prabhumoye, R. Salakhutdinov, A.W. Black, Topological sort for sentence ordering, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 2783–2792.
[21] B. Cui, Y. Li, Z. Zhang, BERT-enhanced relational sentence ordering network, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP, 2020, pp. 6310–6320.
[22] T. Wang, X. Wan, Hierarchical attention networks for sentence ordering, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019.
[23] X. Chen, X. Qiu, X. Huang, Neural sentence ordering, 2016, arXiv preprint arXiv:1607.06952.
[24] S.B.R. Chowdhury, F. Brahman, S. Chaturvedi, Is everything in order? A simple way to order sentences, 2021, arXiv preprint arXiv:2104.07064.
[25] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7871–7880.
[26] D. Batra, A.X. Chang, S. Chernova, A.J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, et al., Rearrangement: A challenge for embodied ai, 2020, arXiv preprint arXiv:2011.01975.
[27] Q.A. Wei, S. Ding, J.J. Park, R. Sajnani, A. Poulenard, S. Sridhar, L. Guibas, Lego-net: Learning regular rearrangements of objects in rooms, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 19037–19047.
[28] T. Caelli, T. Caetano, Graphical models for graph matching: Approximate models and optimal algorithms, Pattern Recognit. Lett. 26 (3) (2005) 339–346.
[29] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, Y. Sun, Masked label prediction: Unified message passing model for semi-supervised classification, in: IJCAI, 2021.
[30] J. Song, C. Meng, S. Ermon, Denoising diffusion implicit models, 2020, arXiv preprint arXiv:2010.02502.
[31] C.-H. Lee, Z. Liu, L. Wu, P. Luo, Maskgan: Towards diverse and interactive facial image manipulation, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2020.
[32] W.R. Tan, C.S. Chan, H. Aguirre, K. Tanaka, Improved ArtGAN for conditional synthesis of natural image and artwork, IEEE Trans. Image Process. 28 (1) (2019) 394–409.
[33] B. Oh, S. Seo, C. Shin, E. Jo, K.-H. Lee, Topic-guided coherence modeling for sentence ordering by preserving global and local information, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, 2019, pp. 2273–2283.
[34] P. Kumar, D. Brahma, H. Karnick, P. Rai, Deep attentive ranking networks for learning to order sentences, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, (05) 2020, pp. 8115–8122.
[35] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: ICML, PMLR, 2019, pp. 6105–6114.
[36] D. Paschalidou, A. Kar, M. Shugrina, K. Kreis, A. Geiger, S. Fidler, Atiss: Autoregressive transformers for indoor scene synthesis, Adv. Neural Inf. Process. Syst. 34 (2021) 12013–12026.
[37] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 652–660.
[38] H. Zheng, Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders, Adv. Neural Inf. Process. Syst. 36 (2022) (2022).
[39] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (7) (2011).