Fragment Relation Networks for Geometric Shape Assembly

Jinhwi Lee[*]
POSTECH, POSCO
jinhwi@postech.ac.kr

Jungtaek Kim[*]
POSTECH
jtkim@postech.ac.kr

Hyunsoo Chung
POSTECH
hschung2@postech.ac.kr

Jaesik Park
POSTECH
jaesik.park@postech.ac.kr

Minsu Cho
POSTECH
mscho@postech.ac.kr

Abstract

A geometric shape is often made of multiple fragments or parts. Assembling the fragments into the target object can be viewed as an interesting combinatorial problem with a variety of applications in science and engineering. Previous related work, however, focuses on tackling limited cases, e.g., primitive fragments of identical shapes or jigsaw-style fragments of textured shapes, which greatly mitigate the combinatorial challenge. In this work we introduce a challenging problem of shape assembly with textureless fragments of arbitrary shapes and propose a learning-based approach to solving it. Given a target object and a set of candidate fragments, the proposed model learns to select one of the fragments and place it into a right place. Our model processes the candidate fragments in a permutation-equivariant manner and can generalize to cases with an arbitrary number of fragments and even with a different target object. We demonstrate our method on shape assembly tasks with different shapes and assembling scenarios.

1 Introduction

There have been a large volume of studies [13] on different types of shape assembly or composition problems in a variety of fields such as biology [17], earth science [19], archaeology [7], and tiling puzzle [14]. If we focus on shapes without textures, the problem can be formulated as a combinatorial optimization problem, which aims to occupy a target object using candidate parts [6], and also be related to the bin packing problem [3], which is one of the representative problems in the combinatorial optimization. Most previous related work, however, tackles shape assembly problems relying on distinct patterns across fragments or parts, e.g., compatible junctions or textures between candidate fragments. For example, a jigsaw puzzle problem finds a correct combination by comparing and matching neighbors of each fragment where the fragments contain visual information [5]. In general cases where such useful hints are not available (i.e., no texture and indistinctive junctions), assembling geometric fragments becomes challenging.

In this work, we introduce a simple yet challenging shape assembly problem in the following scenario. (i) Stochastic partitioning process [16] splits a two-dimensional target object so that their fragments become arbitrary polygons. (ii) The set of fragments are given to reconstruct the target object. (iii) The fragments are supposed to assemble from bottom to top and from left to right; this constraint may lower the degree of difficulty but can render the problem closer to some real environments [4].

[*]Equal contribution.

1st Workshop on Learning Meets Combinatorial Algorithms @ NeurIPS 2020, Vancouver, Canada.

Under the scenario mentioned above, we can solve the shape assembly problem with one of (i) brute-force approach, (ii) metaheuristic [1], (iii) sequential model-based optimization approach [2], and (iv) learning-based approach [12]. Due to the combinatorial explosion, a brute-force approach is infeasible for our purpose when the number of fragments increases. Moreover, both metaheuristic and sequential model-based optimization are not suitable to solve this problem, since such approaches are time-consuming for their inner loop steps. We propose an efficient and effective learning-based approach as an alternative. Given a target object and a set of candidate fragments, the proposed model learns to select one of the fragments and place it into a right place; it processes the candidate fragments in a permutation-invariant manner and can generalize to cases with an arbitrary number of fragments and even with a different target object. We demonstrate our method on shape assembly tasks with different shapes and assembling scenarios.

2 Problem Definition

Geometric Shape Assembly Suppose that we are given a set of fragments $\mathcal{X} = \{\mathbf{x}_j\}_{j=1}^{N}$ and a target object S on a space Φ. In this work we assume polygonal shapes in a two-dimensional space. We sequentially assemble those fragments into the target object; at each step, a fragment $\mathbf{x}_i$ is sampled without replacement and placed on top of the current shape $\mathbf{c}$ considering the target object S. The goal is to build the target object S using all the fragments.

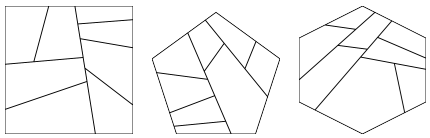


Figure 1: Shape fragmentation examples.

Shape Fragmentation Dataset We create a dataset by partitioning a shape into multiple fragments, which can be used to pose its reverse task, i.e., an assembly problem. Inspired by binary space partitioning algorithm [18], we randomly split a target object and create a set of random fragments for each target object. The procedure is summarized in Algorithm 2 and some examples are shown in Figure 1. After K times of binary partitioning, we obtain $N = 2^K$ fragments.

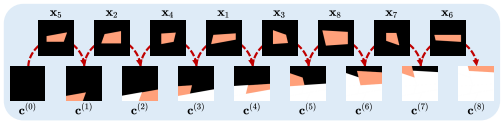


Figure 2: Assembling scenario. A fragment $\mathbf{x}$ is assembled into a shape $\mathbf{c}$. At every step, we select which fragment is assembled and where fragment is assembled.

3 Our Approach

To address the shape assembly problem, we propose a model that learns to select a fragment from current candidates and place it on top of a current shape to build the target object. The model takes two inputs: (i) the remaining shape $S - \mathbf{c}$ for the target object, which is the result of assembled fragments at the previous steps; (ii) the current set of fragments $\mathcal{X}$, which consists of all candidates for the next placement. It then predicts which $\mathbf{x}_i \in \mathcal{X}$ should be used and where it should be placed in Φ. We assemble all the fragments by iteratively running our model until no candidate remains. The procedure is described in Algorithm 1.

To make a proper decision, the model needs to extract geometric information of the fragments and the shape, and also understand the relations among them. We train our model by supervising it with a large number of episodes where a fragment and its position in the next time step is known from a current shape and candidate fragments. We generate such episodes from the shape fragmentation dataset by creating a sequence of fragments from the bottom left to the top right.

Fragment Relation Networks Our model, dubbed the fragment relation network (FRN), considers a candidate fragment $\mathbf{x}_i$ in the context of the other candidates $\mathcal{X} \setminus \mathbf{x}_i$ and the remaining shape $S - \mathbf{c}$, and produces two outputs: (i) the selection probability for $\mathbf{x}_i$, which means how likely $\mathbf{x}_i$ is selected. (ii) the placement probability map for $\mathbf{x}_i$, which means how likely each position is for $\mathbf{x}_i$. As shown in Figure 3, FRN contains two branches for the two outputs, the fragment selection network (i.e., FRN-Sel) and the fragment placement network (i.e., FRN-Loc). Both networks share many learnable

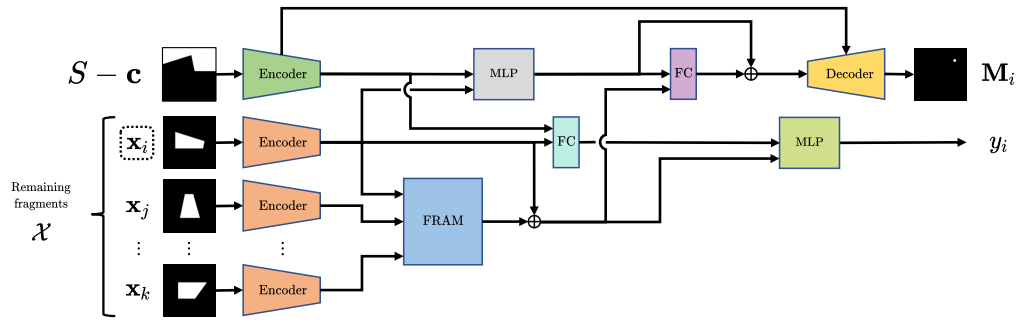


Figure 3: Overall architecture of our model. Given a target object S, a current shape of placed fragments $\mathbf{c}$, and current candidate fragments $\mathcal{X}$, it takes as input the remaining shape $S - \mathbf{c}$ and the current candidate fragments $\mathcal{X}$ and predicts the probabilities of fragments y_i for selection and the probability map $\mathbf{M}_i$ for placement. The fragment encoder (orange) is shared across all fragments.

parameters that are parts of two encoders and fragment relation attention module (FRAM). FRAM is inspired by Transformer network [20] and captures the relationship between the candidate fragments.

Fragment Selection FRN-Sel (the lower part of Figure 3) is a binary classifier taking as input $\mathbf{x}_i, S - \mathbf{c}$, and $\mathcal{X} \setminus \mathbf{x}_i$: $y_i = \text{FRN-Sel}\left(\mathbf{x}_i; S - \mathbf{c}, \mathcal{X} \setminus \mathbf{x}_i\right) \in \mathbb{R}$, where $\mathbf{x}_i \in \mathcal{X}$. Since, in particular, FRN-Sel takes into account fragment relations with FRAM, it can choose the next fragment by considering the remains of fragments. Furthermore, as will be discussed in Appendix D, the number of parameters in FRN does not depend on the number of fragments, which implies that the cardinality of $\mathcal{X}$ can be varied. This property helps our network to handle variable-length set of fragments.

Fragment Placement FRN-Loc (the upper part of Figure 3) determines the position of $\mathbf{x} \in \mathcal{X}$ by predicting a probability map over pixels: $\mathbf{M}_i = \text{FRN-Loc}\left(\mathbf{x}_i; S - \mathbf{c}, \mathcal{X} \setminus \mathbf{x}_i\right) \in \mathbb{R}^{w \times h}$. This network is implemented as an encoder-decoder architecture with skip connections between them, in order to compute pixel-wise probabilities with convolution and transpose convolution operations. It can reduce the number of learnable parameters due to the absence of the last fully-connected layer.

Fragment Relation Attention Module We suggest an attention-based module FRAM, which considers high-order relationship between the remaining fragments using multi-head attention networks [20]. For FRAM, we use multi-head attention and scaled dot-product attention. As proposed by Vaswani et al. [20], given $\mathbf{X} \in \mathbb{R}^{n_1 \times d}$ and $\mathbf{Y} \in \mathbb{R}^{n_2 \times d}$, multi-head attention is composed of multiple scaled dot-product attentions:

$$\text{DP}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \sigma\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_1}}\right)\mathbf{V} \quad \text{and} \quad \text{MH}(\mathbf{X}, \mathbf{Y}, \mathbf{Y}) = [\text{DP}_1, \ldots, \text{DP}_h]\mathbf{W}^O, \tag{1}$$

where $\mathbf{Q} \in \mathbb{R}^{n_1 \times d_1}$, $\mathbf{K} \in \mathbb{R}^{n_2 \times d_1}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times d_2}$, $\text{DP}_i = \text{DP}(\mathbf{X}\mathbf{W}_i^{\mathbf{Q}}, \mathbf{Y}\mathbf{W}_i^{\mathbf{K}}, \mathbf{Y}\mathbf{W}_i^{\mathbf{V}})$, σ is a softmax function, h is the number of heads, and $\mathbf{W}^O \in \mathbb{R}^{hd_2 \times d}$ is the learnable parameters. To improve the expression power, $\mathbf{X}$ and $\mathbf{Y}$ are projected by the different parameter sets $\mathbf{W}^{\mathbf{Q}} \in \mathbb{R}^{d \times d_1}$, $\mathbf{W}^{\mathbf{K}} \in \mathbb{R}^{d \times d_1}$, and $\mathbf{W}^{\mathbf{V}} \in \mathbb{R}^{d \times d_2}$.

To express a set of the remaining fragments to a latent representation $\mathbf{h}$, we leverage the multi-head attention: $\mathcal{H} = \text{MH}(\mathcal{X}, \mathcal{X}, \mathcal{X}) \in \mathbb{R}^{|\mathcal{X}| \times d}$ and $\mathbf{h} = \text{MH}(\mathbf{1}, \mathcal{H}, \mathcal{H}) \in \mathbb{R}^d$, where $\mathbf{1} \in \mathbb{R}^{1 \times d}$ is an all-ones matrix. Without loss of generality, the aforementioned equations can take $\mathcal{X}$ and $\mathcal{H}$, respectively. Moreover, $\text{MH}(\mathcal{X}, \mathcal{X}, \mathcal{X})$ can be stacked by feeding $\mathcal{H}$ to the multi-head attention as an input, and $\text{MH}(\mathbf{1}, \mathcal{H}, \mathcal{H})$ can be also stacked in the similar manner. Note that $\mathbf{h}$ becomes $\mathbb{R}^{1 \times d}$, but we straightforwardly employ it as a $\mathbb{R}^d$-shaped representation.

FRAM is a generalization of global feature aggregation methods such as average pooling and max pooling across instances, so that it allows high-order interaction between instances and aggregates instance-wise representations (i.e., $\mathcal{H}$) with learnable parameters. The final output is determined by

(a) `Square` (b) `Pentagon` (c) `Hexagon`

Figure 4: Shape assembly results for `Square`, `Pentagon` and `Hexagon` shape. The orange lines delineate target objects. For each target shape, the results of simulated annealing (SA), Bayesian optimization (BO), V-GAN (only for `Square`), and ours are compared.

selecting the fragment with the index of maximum y_i: $i^\star = \arg\max_{i \in \{1,\ldots,|\mathcal{X}|\}} y_i$. After choosing the fragment, the position with the maximum probability on $\mathbf{M}_{i^\star}$ is selected for the placement.

4 Experimental Results

Experimental Setup As shown in Figure 1, we evaluate our method and the existing methods on three types of target objects: (i) `Square`, (ii) `Pentagon`, and (iii) `Hexagon`. Unless otherwise specified, we use 5,000 examples each of them is partitioned into 8 fragments using binary space partitioning. We use 64%, 16%, and 20% of the examples for training, validation, and test datasets, respectively.

Note that there is no prior methods that tackle shape assembly for textureless fragments. Therefore, we apply two types of classic optimization methods for this task. Besides, we apply an approach that is based on generative adversarial networks. Due to a page limit, the architectures of FRN and V-GAN will be presented in the appendices. We measure an assembling quality via IoU: $\text{IoU}(\mathbf{c}, S) = \frac{\text{area}(\mathbf{c} \cap S)}{\text{area}(\mathbf{c} \cup S)}$, where $\mathbf{c}$ is an assembled shape and S is a target object.

Main Result We compare our approach to other baseline methods on the assembly of different target objects. Results on `Square`, `Pentagon`, and `Hexagon` shapes are summarized in Table 1.

V-GAN requires a fixed number of vertices for partitioned fragments, which is only applicable to `Square` target shape. We find that our method consistently covers much more area than baselines within less time budget. Both simulated annealing and Bayesian optimization struggle to place fragments in the proper position, resulting in many overlaps between fragments. They also take much longer time compared to our method in all experimental conditions. On the other hand, V-GAN takes much less time than other baseline methods while significantly underperforms compared to other methods. Qualitative results are presented in Figure 4.

Table 1: Quantitative results on three shapes.

		mIoU	**Time**
`Square`	SA	0.8524	2,700 sec.
	BayesOpt	0.8329	1,180 sec.
	V-GAN	0.5615	**< 1 sec.**
	Ours	**0.8938**	**< 1 sec.**
`Pentagon`	SA	0.8349	2,584 sec.
	BayesOpt	0.7812	1,240 sec.
	V-GAN	N/A	
	Ours	**0.9262**	**< 1 sec.**
`Hexagon`	SA	0.8230	2,520 sec.
	BayesOpt	0.7745	1,210 sec.
	V-GAN	N/A	
	Ours	**0.9323**	**< 1 sec.**

5 Conclusion

In this paper, we solve a two-dimensional shape assembly problem with our proposed neural network FRN. It predicts the next fragment and its corresponding position where the fragments that would be assembled are given, by considering fragment relations with an attention-based module FRAM. We show that our method outperforms other baseline methods such as simulated annealing, Bayesian optimization, and a simple learning-based approach with adversarial training, in the circumstances each of which deals with one of three different target geometric shapes.

Acknowledgments and Disclosure of Funding

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH)).

References

[1] I. Boussaïd, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

[2] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[3] A. R. Brown. *Optimum packing and depletion*. Macdonald and Co.; New York, American Elsevier, 1971.

[4] B. Chazelle. The bottomn-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, C-32(8):697–707, 1983.

[5] T. S. Cho, S. Avidan, and W. T. Freeman. A probabilistic image jigsaw puzzle solver. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 183–190, 2010.

[6] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. *Approximation Algorithms for NP-hard Problems*, pages 46–93, 1996.

[7] N. Derech, A. Tal, and I. Shimshoni. Solving archaeological puzzles. *arXiv preprint arXiv:1812.10553*, 2018.

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 27, pages 2672–2680, 2014.

[9] N. Guttenberg, N. Virgo, O. Witkowski, H. Aoki, and R. Kanai. Permutation-equivariant neural networks applied to dynamics prediction. *arXiv preprint arXiv:1612.04530*, 2016.

[10] D. P. Kingma and J. L. Ba. ADAM: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[11] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3744–3753, 2019.

[12] J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu. LayoutGAN: Synthesizing graphic layouts with vector-wireframe adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[13] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002.

[14] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 69–84, 2016.

[15] M. Pincus. Letter to the editor – a Monte Carlo method for the approximate solution of certain types of constrained optimization problems. *Operations research*, 18(6):1225–1228, 1970.

[16] D. M. Roy and Y. W. Teh. The Mondrian process. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 21, pages 1377–1384, 2008.

[17] C. A. A. Sanches and N. Y. Soma. A polynomial-time DNA computing solution for the bin-packing problem. *Applied Mathematics and Computation*, 215(6):2055–2062, 2009.

[18] R. Schumacher. Study for applying computer-generated images to visual simulation. Technical Report AFHRL-TR-69-14, Air Force Human Resources Laboratory, Air Force Systems Command, 1969.

[19] T. H. Torsvik. The Rodinia jigsaw puzzle. *Science*, 300(5624):1379–1381, 2003.

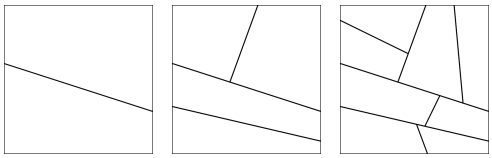
[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5998–6008, 2017.

[21] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 3391–3401, 2017.

Appendices

In this material, we describe the detailed contents that supplement our main article.

Appendix A Details of Shape Fragmentation



(a) Partitioning 1 (b) Partitioning 2 (c) Partitioning 3

Figure 5: Fragmentation examples on `Square` by the number of partitions.

We divide a target geometric shape into fragments using hyperplanes by a binary space partitioning algorithm [18]. The number of fragments increases exponentially with the number of partitions. Some partitioning examples are shown in Figure 5.

We create a shape fragmentation dataset under two constraints that prevent generating small fragments. First, the hyperplane does not cross adjacent edges of the given polygon. Second, the position that the hyperplane will pass through is randomly selected between the range of $\pm$ 25% of edge length from the center of the edge. The dataset we used is composed of 5000 samples.

Appendix B Algorithms

The algorithms described in the main article are represented in Algorithm 1 and Algorithm 2.

Algorithm 1 Geometric Shape Assembly Procedure

Input: fragments $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ and a target object S
Output: output shape $\mathbf{c}$.

Initialize current shape $\mathbf{c}$ (as null) on a space Φ.
Initialize remaining shape $S - \mathbf{c}$
while $\mathcal{X} \neq \emptyset$ **do**
  Select a fragment $\mathbf{x}_i \in \mathcal{X}$ and place it on Φ.
  $\mathbf{c} \leftarrow$ shape $\mathbf{c}$ updated by the placement of $\mathbf{x}_i$.
  Calculate the remaining shape $S - \mathbf{c}$.
  $\mathcal{X} \leftarrow \mathcal{X} \setminus \mathbf{x}_i$
end while
return $\mathbf{c}$

Algorithm 2 Shape Fragmentation

Input: A target object S, the number of partitions K.
Output: Fragments partitioned $\mathcal{X}_K = \{\mathbf{x}_i\}_{i=1}^{2^K}$.

function partitionFragment($\mathbf{x}$)
  Choose where to partition.
  Partition $\mathbf{x}$ to $\mathbf{x}_+, \mathbf{x}_-$.
  return $\mathbf{x}_+, \mathbf{x}_-$
end function

$\mathcal{X}_0 = \{S\}$.
for $i = 1, \ldots, K$ **do**
  $\mathcal{X}_i = \{\}$
  for all $\mathbf{x} \in \mathcal{X}_{i-1}$ **do**
    $\mathbf{x}_+, \mathbf{x}_- =$ partitionFragment($\mathbf{x}$)
    $\mathcal{X}_i \leftarrow \mathcal{X}_i + \{\mathbf{x}_+, \mathbf{x}_-\}$
  end for
end for
return a set of fragments $\mathcal{X}_K$

Appendix C Details of Experiment

C.1 Training

We alternately train our neural network with the following objectives. Since we first choose the next fragment, and then place the selected fragment, the gradient updates by two objectives should be separately applied. The objective for the fragment selection part is

$$\mathcal{L}_{\text{select}} = \sum_{m=1}^{M} -\mathbf{t}_m^\top \log \sigma(\mathbf{y}_m), \tag{2}$$

where $\mathbf{t}_m$ is a one-hot representation of true fragment, each entry of $\mathbf{y}_m$ is computed by FRN-Sel, and σ is a softmax function.

Next, the objective for the fragment placement part is

$$\mathcal{L}_{\text{place}} = \sum_{m=1}^{M} -\text{vector}(\boldsymbol{\tau}_m)^\top \log \text{vector}(\mathbf{M}_m), \tag{3}$$

where $\boldsymbol{\tau}_m \in \mathbb{R}^{w \times h}$ is a one-hot representation of true position, $\mathbf{M}_m$ is computed by FRN-Loc, and vector is a function for vectorizing a matrix.

Both Eq. (2) and Eq. (3) compute the summation of the cross-entropy over outputs that produce through a permutation-equivariant network for logits. As will be described in the subsequent sections, this property let us examine FRN in various circumstances. Moreover, since our network does not rely on the sequence (or history) of fragments, we are able to apply a mini-batch training scheme, which enables us to train offline.

C.2 Baselines

We select the following three baseline methods:

Simulated Annealing (SA) [15] It solves the problem via optimization. SA computes the overlap to a target object;

Bayesian Optimization (BayesOpt) [2] A powerful method to optimize a black-box function. We find the position of given fragments to maximize the overlap between each fragment and target object;

LayoutGAN Modified for Vertex Inputs (V-GAN) [12] It is a modification of [12]. Since LayoutGAN assumes that fragment shapes are always fixed and only finds their positions, we modify the generator structure to a neural network that takes fragment vertices and predicts their positions. This baseline is composed of the generator, and the discriminator that distinguishes real and fake examples is trained via an adversarial training strategy.

C.3 Details of Model Architecture

FRN FRN consists of FRN-Sel that selects next fragment and FRN-Loc that predicts center coordinates of the chosen fragment. There exist 7 hyperparameters in FRN with 3 specifically in FRAM. The default dimension of fully connected layer in the whole pipeline is fixed to 256 throughout the experiments. For the training, we use batch size of 32 and ADAM optimizer [10] for both FRN-Sel and FRN-Loc with learning rate of 5×10^{-4} and 2×10^{-3} respectively. FRN-Sel is a convolutional neural network (CNN) followed by FRAM and MLP block. FRN-Loc is convolutional encoder-decoder with MLP block followed by an additional fully-connected layer inserted in between. Each encoder for FRN-Loc and FRN-Sel do not share the weights, and thus are trained with different loss. Our FRAM follows similar setting of original Transformer [20] model with 2048 hidden units and 8 attention heads.

V-GAN It follows typical setup of generative adversarial network [8], having both generator and discriminator. Specifically, generator takes fragment's vertex coordinates as input and predicts center coordinates. Discriminator then makes real or fake decision based on adjusted vertex coordinates. For training, we use batch size of 128 and ADAM optimizer with learning rate of 1×10^{-4}.

Appendix D Discussion and Related Work

In this section, we discuss the properties of our problem and FRN, based on the numerical results. Moreover, we mention about the previous studies related to our work.

As presented in Table 1, the coverage result that assembles `Hexagon` is better than the results for `Square` and `Pentagon`. It implies that if a target object becomes complicated and contains more information, the problem becomes easily solvable. From this observation, we can readily induce our formulation to the packing problem with inherent information such as [5] and [14].

Our proposed network FRN can be thought as a permutation-equivariant network, which satisfies

$$f(\pi([\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top)) = \pi[f([\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top)], \tag{4}$$

where $f : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d'}$ is a neural network that outputs a set and π is a permutation function along the first dimension. Guttenberg et al. [9] suggest a permutational layer to handle the different inputs and not depend on the specific ordering of inputs. Zaheer et al. [21] derive the necessary and sufficient conditions on permutation-equivariant networks. Lee et al. [11] propose an attention-based permutation-equivariant block for set operations.

By this definition of permutation-equivariance, we can describe the following proposition for FRN:

Proposition 1 *By the shared architecture over the remaining fragments $\mathcal{X}$, FRN is permutation-equivariant. In particular, the number of learnable parameters does not depend on the dimensionality of logits $\mathbf{y}$ and the shape of position probability $\mathbf{M}_i$ for $i \in \{1, \ldots, |\mathcal{X}|\}$.*

As computed by FRN-Sel and FRN-Loc, their outputs can handle variable-sized $\mathcal{X}$ and the ordering of $\mathcal{X}$ affects the ordering of the outputs, satisfying Eq. (4). The experimental results support that our method can learn this challenging scenario with the permutation-equivariant neural network.