A Code of our method

The code of our paper is attached in the `CodeForReview` directory.

B WeaveNet and its Extension for Sparse Bipartite Graphs

WeaveNet (WN) is a network written as a function $\mathcal{M} : \mathbf{Z} \mapsto \boldsymbol{M}$, where $\mathbf{Z} \in \mathbb{R}^{N,M,D}$ is D-dimensional attributes of $N \times M$ edges. $\boldsymbol{M} \in \{0,1\}^{N \times M}$ is an estimate of matching in the form of a binary matrix.

WN is originally developed to solve stable matching approximately. Stable matching is a *strongly NP-hard* problem defined on a bipartite graph. Let us consider a bipartite graph that has N nodes on one side and M nodes on the opposite side, which we refer to as $\mathcal{P}$ and $\mathcal{Q}$ on this problem, respectively ($|\mathcal{P}| = N$ and $|\mathcal{Q}| = M$). The graph has $N \times M$ edges. The input of stable matching differs from the well-known linear assignment problem; the input is weights on directed edges, where directions are $\mathcal{P} \to \mathcal{Q}$ and $\mathcal{Q} \to \mathcal{P}$. We can represent edge weights by a matrix of the size $N \times M$. Hence, for two directions, we have $\boldsymbol{P} \in \mathbb{R}^{N \times M}$ and $\boldsymbol{Q} \in \mathbb{R}^{M \times N}$ edge weights.

From these two matrices, $\mathbf{Z} \in \mathbb{R}^{N \times M \times 2}$ is obtained as

$$\mathbf{Z} = \mathrm{cat}(\boldsymbol{P}, \boldsymbol{Q}^\top). \tag{9}$$

We focused on the fact that these inputs and outputs fit to our intention shown in (2). In addition, we expected that the network-based algorithm is innately general-purposed and can solve our problem of stochastic linear assignment under an unknown distribution shape. The results reported in 4 proved the expectation was correct.

B.1 Details of the WeaveNet architecture

WN comprises L feature weaving layers followed by one output layer. Let $(\mathbf{Z}_\ell^{\mathcal{P}}, \mathbf{Z}_\ell^{\mathcal{Q}})$ be the input to ℓ-th layer. We describe the ℓ-th feature weaving layer as a function $f_\ell : (\mathbf{Z}_\ell^{\mathcal{P}}, \mathbf{Z}_\ell^{\mathcal{Q}}) \mapsto (\mathbf{Z}_{\ell+1}^{\mathcal{P}}, \mathbf{Z}_{\ell+1}^{\mathcal{Q}})$. The input for the first layer is given as $(\mathbf{Z}_0^{\mathcal{P}}, \mathbf{Z}^{\mathcal{Q}}) = (\mathbf{Z}, \mathrm{swap}(\mathbf{Z}))$, where swap is a function that swap the components derived from $\boldsymbol{P}$ and $\boldsymbol{Q}$ in the tensor $\mathbf{Z}$ (i.e., the result is identical to $\mathbf{Z}_0^{\mathcal{Q}} = \mathrm{cat}(Q, P^\top)$ for Eq. (9), or $\boldsymbol{z}_{(m,n)}^{\mathcal{Q}} = \mathrm{cat}(\boldsymbol{q}_m^2, d(\boldsymbol{q}_m^1, \boldsymbol{p}_n^1), \boldsymbol{p}_n^2)$ for Eq. (8)).

f_ℓ is designed to pass messages frequently among neighbor nodes on a graph. Hereafter, we only explain the calculation for the $\mathcal{P} \to \mathcal{Q}$ direction for simplicity. Let $\mathcal{N}(p_n)$ be a set of neighbors of node p_n. On a bipartite graph, $\mathcal{N}(p_n) = \mathcal{Q}$ and $\mathcal{N}(q_m) = \mathcal{P}$. The uniqueness of each matching candidate of p_n should be emphasized for matching since selecting only one partner among many candidates is the task. A feature weaving layer uses the calculation originally proposed for point segmentation. Namely, it describes the characteristic of q_m among $\mathcal{Q}$ as

$$\boldsymbol{h}_{\ell,n} = \mathrm{max\_pooling}\{\phi_\ell^1(\boldsymbol{z}_{\ell,(n,m)}) | q_m \in \mathcal{N}(p_n)\}, \tag{10}$$

where max_pooling is the max pooling operation, ϕ_ℓ^1 is a linear layer inside of f_ℓ, and $\boldsymbol{z}_{(n,m,\ell)}$ is (n,m)-th element in $\mathbf{Z}_\ell^{\mathcal{P}}$. Here, $\mathbf{h}_{\ell,n}^A$ represent a group characteriztic of $\mathcal{N}(p_n)$. $\boldsymbol{h}_{\ell,n}$ is compared with individual edge features $\boldsymbol{z}_{\ell,(n,m)}$ as

$$\boldsymbol{g}_{\ell,(n,m)} = \mathrm{pReLU}(\mathrm{BN}(\phi_\ell^2(\mathrm{cat}(\boldsymbol{z}_{\ell,(n,m)}, \boldsymbol{h}_{\ell,n})))), \tag{11}$$

where ϕ_ℓ^2 is another linear layer inside of f_ℓ, pReLU is the pReLU function, and BN is the batch normalization operation.

Second, the layer mixes features obtained for each side to pass messages each other. Let $\boldsymbol{g}_{\ell,(n,m)}^{\mathcal{P}}$ and $\boldsymbol{g}_{\ell,(m,n)}^{\mathcal{Q}}$ be vectors obtained by Eq. (11) for each direction. f_ℓ concatenate them at the end of calculation, which yields

$$\boldsymbol{z}_{\ell+1,(n,m)}^{\mathcal{P}} = \mathrm{cat}(\boldsymbol{g}_{\ell,(n,m)}^{\mathcal{P}}, \boldsymbol{g}_{\ell,(m,n)}^{\mathcal{Q}}). \tag{12}$$

Finally, the L-th output $\boldsymbol{z}_{L+1,(n,m)}$ is fed to the output layer, which first applies Eq. (10) with $\boldsymbol{z}_{L+1}$ as its input, then, calculate

$$g_{n,m} = \mathrm{softmax}(BN(\phi_{output}(\mathrm{cat}(\boldsymbol{z}_{L+1,(n,m)}, \boldsymbol{h}_{L+1,(n,m)})))), \tag{13}$$

where softmax is the softmax function, and $g_{n,m}$ is (n,m)-the element in $\boldsymbol{M}$. Note that we obtain the estimate for both directions, and use their average for prediction.

B.2 Extension for Sparse Bipartite Graph

Unlike the stable matching problem, we can prune edges based on the distance matrix in advance. We have re-implemented WeaveNet using `torch-geometric` to support such edge pruning. It also causes some differences in mathematical notation from those given in the previous subsection. First, in (10), $\mathcal{N}(p_n) = \mathcal{Q}$ in the original paper, but $\mathcal{N}(p_n) = \{q_m | d(p_n, q_m) \leq r\}$ in our version. Second, we set $g_{n,m}$ depending on whether $(n,m) \in \boldsymbol{E}$ or not, as

$$g'_{n,m} = \begin{cases} g_{n,m} & \text{if } (n,m) \in \boldsymbol{E} \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

C Mathematical definition of evaluation metrics

Evaluation metrics for the 4DMatch, 4DLoMatch, 3DMatch, and 3DLoMatch. We used the inlier ratio (IR) (also known as accuracy [23]), and non-rigid feature matching recall (NFMR) as the evaluation metrics for the 4DMatch and 4DLoMatch datasets following [22, 23, 51]. For ground-truth matches ($\mathbf{u} \in \mathcal{R}^{\mathbf{3}}$,$\mathbf{v} \in \mathcal{R}^{\mathbf{3}}$) $\in \mathcal{K}_{\mathrm{gt}}$, predicted matches ($\hat{\mathbf{u}} \in \mathcal{R}^{\mathbf{3}}$,$\hat{\mathbf{v}} \in \mathcal{R}^{\mathbf{3}}$) $\in \mathcal{K}_{\mathrm{pred}}$, and the

ground-truth warping function W_{gt}, IR, and NFMR are defined as follows:

$$\text{IR} = \frac{1}{|\mathcal{K}_{\text{pred}}|} \sum_{(\hat{\mathbf{u}},\hat{\mathbf{v}})\in\mathcal{K}_{\text{pred}}} [||W_{\text{gt}}(\hat{\mathbf{u}}) - \hat{\mathbf{v}}||_{\mathbf{2}} < \sigma], \tag{15}$$

$$\text{NFMR} = \frac{1}{|\mathcal{K}_{\text{gt}}|} \sum_{(\mathbf{u},\mathbf{v})\in\mathcal{K}_{\text{gt}}} [||\Gamma(\mathbf{u},\mathbf{A},\mathcal{F}) - \mathbf{v}||_{\mathbf{2}} < \sigma], \tag{16}$$

$$\Gamma(\mathbf{u},\mathbf{A},\mathcal{F}) = \sum_{\text{A}_i\in\text{knn}(\mathbf{u},\mathbf{A})} \frac{F_i||\hat{\mathbf{u}} - \mathbf{A}_i||_{\mathbf{2}}^{-\mathbf{1}}}{\sum_{\text{A}_i\in\text{knn}(\mathbf{u},\mathbf{A})} ||\mathbf{u} - \mathbf{A_i}||_{\mathbf{2}}^{-\mathbf{1}}}, \tag{17}$$

$$\mathcal{F} = \{\hat{\mathbf{v}} - \hat{\mathbf{u}} | (\hat{\mathbf{u}},\hat{\mathbf{v}}) \in \mathcal{K}_{\text{pred}}\},\ \text{A} = \{\hat{\mathbf{u}} | (\hat{\mathbf{u}},\hat{\mathbf{v}}) \in \mathcal{K}_{\text{pred}}\}, \tag{18}$$

where σ is set as 0.04 m, and $||\cdot||_2$ and $[\cdot]$ represent the L2-norm and the Iverson bracket, respectively.

The IR, feature matching recall (FMR), and rigid registration recall (RR) are used as the evaluation metrics for the 3DMatch and 3DLoMatch datasets following [22]. The FMR indicates the fraction of pairs with >5% inlier matches with <10 cm residual under the ground truth transformation, and the RR indicates the fraction of scan pairs where the correct transformation parameters are identified with RANSAC.

Evaluation metrics for Human Shape data . Following [54], we used the corresponding percentage (Corr) under controlled error tolerances as the evaluation metric for Human Shape data. The Corr and error tolerance are defined as follows:

$$\text{Corr} = \frac{1}{N}||\boldsymbol{P} \odot \boldsymbol{P}^{\text{gt}}||_1,\ \text{error tolerance} = r/\max\{||\boldsymbol{p}_n - \boldsymbol{p}_{n'}||, {}^{\forall}n, n'\}, \tag{19}$$

where $\odot$ and $||\cdot||_1$ represent the Hadamard product and L1-norm, respectively. In addition, $\boldsymbol{P}^{\text{gt}}$ and r represent the ground-truth correspondence matrix and the tolerant radius, respectively.

D Additional Reports on Memory Consumption

An analysis of memory consumption, measured in actual training and inference on 4DMatch with RoITr, is shown in Sec. 4.2 and Tab. 3 of the main paper. Here, we show additional results on 4DMatch with Lepard and LNDP in Tab. 5 and Tab. 6. We also privide that on 3DMatch in Tab. 7, Tab. 8, and Tab. 9. These results show that our modification reduces the memory consumption without performance drop, regardless of the dataset and method differences.

Table 5: Memory consumption test on 4DMatch with Lepard. ES and FS stand for Edge Selection and Feature Summarization, respectively.

Method	ES	FS	train	eval.	NFMR(↑)	IR(↑)
WN			151.5 GiB	80.5 GiB	**91.2**	83.2
		✓	143.2 GiB	72.1 GiB	<u>90.1</u>	84.2
	✓		18.7 GiB	10.4 GiB	89.5	<u>85.4</u>
	✓	✓	13.4 GiB	7.6 GiB	86.7	**86.1**
DS	-	-	6.5 GiB	3.5 GiB	83.7	82.7

Table 6: Memory consumption test on 4DMatch with LNDP.

Method	ES	FS	train	eval.	NFMR(↑)	IR(↑)
WN			148.2 GiB	68.7 GiB	**91.3**	85.4
		✓	137.4 GiB	68.5 GiB	<u>90.3</u>	85.5
	✓		16.5 GiB	8.8 GiB	89.5	<u>86.4</u>
	✓	✓	10.3 GiB	5.8 GiB	88.7	**87.9**
DS	-	-	4.9 GiB	2.5 GiB	85.4	84.5

Table 7: Memory consumption test on 3DMatch with Lepard.

Method	ES	FS	train	eval.	FMR(↑)	IR(↑)	RR(↑)
WN			169.5 GiB	90.0 GiB	**99.0**	58.1	94.0
		✓	154.3 GiB	84.3 GiB	98.5	59.9	<u>94.1</u>
	✓		23.2 GiB	12.3 GiB	<u>98.6</u>	<u>60.3</u>	<u>94.1</u>
	✓	✓	12.7 GiB	6.9 GiB	98.4	**64.5**	**95.7**
DS	-	-	7.0 GiB	4.5 GiB	98.3	55.5	93.5

Table 8: Memory consumption test on 3DMatch with LNDP.

Method	ES	FS	train	eval.	FMR(↑)	IR(↑)	RR(↑)
WN			164.3 GiB	75.6 GiB	**99.0**	59.9	93.0
		✓	145.3 GiB	76.5 GiB	<u>98.9</u>	62.1	93.1
	✓		19.3 GiB	9.4 GiB	<u>98.9</u>	<u>63.5</u>	<u>93.4</u>
	✓	✓	10.3 GiB	6.4 GiB	98.6	**65.6**	**94.1**
DS	-	-	6.7 GiB	3.4 GiB	98.1	56.5	92.4

Table 9: Memory consumption test on 3DMatch with RoITr.

Method	ES	FS	train	eval.	FMR(↑)	IR(↑)	RR(↑)
WN			143.2 GiB	69.4 GiB	**99.1**	83.2	93.4
		✓	132.1 GiB	61.4 GiB	<u>98.9</u>	83.8	94.9
	✓		17.6 GiB	8.9 GiB	<u>98.9</u>	<u>84.8</u>	<u>95.5</u>
	✓	✓	9.9 GiB	5.4 GiB	<u>98.9</u>	**86.8**	**96.2**
OT	-	-	5.8 GiB	2.1 GiB	98.5	80.3	91.0

Table 10: Study on the impact of hyperparameters with Lepard. We set uncontrolled hyperparameters to our defaults (e.g., when r is varied, we used $L = 10$ and $C^2 = 16$).

Method		4DLoMatch		3DLoMatch		
		NFMR(↑)	IR(↑)	FMR(↑)	IR(↑)	RR(↑)
DS		66.9	55.7	84.5	26.0	69.0
WN	$r = 0.1$	68.2	54.1	80.4	25.6	63.5
WN	$r = 0.5$	**75.3**	**72.4**	**89.6**	30.4	**74.9**
WN	$r = 1.0$	69.3	58.9	88.6	**30.6**	74.0
WN	$L = 6$	67.7	57.1	86.2	25.7	70.2
WN	$L = 8$	68.7	57.2	87.2	27.8	73.5
WN	$L = 10$	**72.4**	**62.5**	**89.6**	**30.4**	**74.9**
WN	$C^2 = 4$	69.8	58.4	84.8	26.6	70.9
WN	$C^2 = 16$	**72.4**	**62.5**	**89.6**	**30.4**	**74.9**
WN	$C^2 = 64$	70.1	57.6	87.1	30.2	72.1

Table 11: Study on the impact of hyperparameters with LNDP.

Method		4DLoMatch		3DLoMatch		
		NFMR(↑)	IR(↑)	FMR(↑)	IR(↑)	RR(↑)
DS		67.6	57.6	83.1	27.4	71.1
WN	$r = 0.1$	70.3	58.9	87.1	27.9	73.1
WN	$r = 0.5$	**73.4**	**62.8**	**91.3**	**33.3**	**76.2**
WN	$r = 1.0$	71.1	60.0	88.3	30.6	72.7
WN	$L = 6$	69.3	60.1	87.6	29.0	74.3
WN	$L = 8$	71.3	62.2	90.1	32.3	75.8
WN	$L = 10$	**73.4**	**62.8**	**91.3**	**33.3**	**76.2**
WN	$C^2 = 4$	69.1	58.9	84.9	31.9	72.8
WN	$C^2 = 16$	**73.4**	**62.8**	**91.3**	**33.3**	**76.2**
WN	$C^2 = 64$	70.9	60.9	89.2	32.1	74.7

E Additional Reports on Hyperparameter Validation

An ablation study on hyperparameter validation is shown in Sec. 4.4 and Tab. 4 of the main paper. We also show Lepard and LNDP results in Tab. 10 and Tab. 11. These show that the selected hyperparameter setting works best, regardless of the dataset and method differences.