MatchMakerNet: Enabling Fragment Matching for Cultural Heritage Analysis

Ariana M. Villegas-Suarez
Universidad de Ingenieria y Tecnologia – UTEC
Lima, Peru
ariana.villegas@utec.edu.pe

Cristian Lopez
Universidad de Ingenieria y Tecnologia – UTEC
Lima, Peru
clopezd@utec.edu.pe

Ivan Sipiran
Department of Computer Science, University of Chile
Santiago, Chile
isipiran@dcc.uchile.cl

Abstract

Automating the reassembly of fragmented objects is a complex task with applications in cultural heritage preservation, paleontology, and medicine. However, the matching subtask of the reassembly process has received limited attention, despite its crucial role in reducing the alignment search space. To address this gap, we propose MatchMakerNet, a network architecture designed to automate the pairing of object fragments for reassembly. By taking two point clouds as input and leveraging graph convolution alongside a simplified version of DGCNN, MatchMakerNet achieves remarkable results. After training on the Artifact (synthetic) dataset, we achieve an accuracy of 87.31% in all-to-all comparisons between the fragments. In addition, it demonstrates robust generalization capabilities, achieving 86.93% accuracy on the Everyday (synthetic) dataset and 83.03% on the Puzzles 3D (real-world) dataset. These findings highlight the effectiveness and versatility of MatchMakerNet in solving the matching subtask.

1. Introduction

Preserving our cultural heritage involves reassembling together fragmented artifacts to create a meaningful representation of the past. However, this task is often complex and time-consuming. The challenge of reassembling broken objects is not exclusive to cultural heritage [24, 34, 3, 20, 36, 10, 40]; it extends to fields like paleontology [19, 16, 28] and medicine [13, 35, 14, 1, 31], where the reconstruction of fractured bones is essential for accurate diagnosis and effective treatment planning. Advancements in 3D object scanning [7, 8, 4] have motivated researchers to explore computer vision and machine learning for automating the process of 3D object reassembly. This shift has revolutionized the way objects are reconstructed across various domains [19, 16, 35].

Reassembly involves various subtasks depending on the specific adopted approach. These subtasks include point cloud feature extraction [25, 37, 22, 9, 26], sampling [18, 17, 21, 23, 38], matching [39, 34, 11, 44, 2], and alignment [19, 5, 13, 15, 3]. In some cases, the matching step may be optional if the method is capable of handling the alignment of all fragments simultaneously. However, the efficacy of this approach is constrained when dealing with intricate fracture patterns characterized by high levels of detail, commonly referred to as complex fractures [11, 43]. As a result, reassembling high detailed fracture faces can exponentially increase the search space, posing computational challenges and potentially compromising the accuracy.

Therefore, a matching step is crucial for effectively reducing the alignment search space. Considering that the matching step requires the evaluation of all potential pairs of fragments, we need to devise an efficient approach for this task. This becomes particularly important when working with real-world datasets that contain complex fracture patterns, such as the Puzzles 3D dataset [12]. By incorporating a matching step, the reassembly process can achieve a focused search, leading to improved efficiency and accurate alignment.

To the best of our knowledge, previous methods have not thoroughly explored the matching subtask. Some methods consider matching to be a trivial process, focusing solely on the final reassembly result [15, 20, 44, 2, 10, 40]. However, this assumption overlooks the challenges posed by 3D objects with complex fractures, as seen in the Breaking Bad dataset [29]. Furthermore, the evaluation of matching is often limited to small and private datasets [39, 34], making it difficult to compare results with newer matching methods.

In this work, we focus on thick fragments, which have more complex fracture surfaces and have received less attention in the state-of-the-art literature. Drawing inspiration from graph convolution [37] and vector neurons [6], we propose a simplified version of DGCNN as our feature extractor. Additionally, we introduce MatchMakerNet to discriminate between pairs of thick fragments sharing at least one fractured face. By leveraging the properties of graph convolution, MatchMakerNet addresses the challenges of

reassembly with complex fracture patterns.

We evaluate our simplified DGCNN alongside other state-of-the-art methods in terms of accuracy, memory usage, and inference time for classification on ModelNet40 [27]. Besides, we assess the performance of MatchMakerNet on both synthetic data from Breaking Bad [29] and real fragmented 3D objects from Puzzles 3D [12]. In these experiments, we compared the loss, accuracy and F1 score, providing an analysis of our method's effectiveness in handling different scenarios involving fragmented 3D objects.

Contribution. We propose a simplified version of DGCNN that maintains a performance of approximately 87% accuracy while significantly reducing the inference time by around 2.82 times and minimizing memory costs by approximately 8.69 times. Additionally, we introduce MatchMakerNet, a network architecture that effectively handles the matching subtask and achieves an accuracy of 87% on the Artifact subset. We also analyze the generalization capabilities of our approach on diverse datasets, obtaining accuracy rates of 87% and 83% on Everyday subset and Puzzles 3D, respectively.

2. Related Work

In this section, we provide a concise review of reassembly methods with different approaches, with a special focus on matching subtask, and datasets related to 3D fragmented objects.

3D object reassembly. Existing methods for 3D object reassembly focus on using either fractured faces [39, 19, 34] or complete fragments [3, 11, 43] as inputs. When the input is a fracture face, an additional step is required to extract these faces from the objects. In addition, some reassembly methods handle objects consisting of only two fragments [19, 13, 15, 3, 30], while others are capable of handling scenarios involving multiple fragments [39, 34, 10, 43]. In the latter case, both two-step [39, 34, 44, 10, 40] and single-step [11, 43] approaches can be employed for the reassembly process.

The two-step approach divides the reassembly process into two subtasks. First, the matching step identifies pairs of fragments that should be reassembled together [39, 34, 10]. Second, the alignment step is carried out to reassemble the fragments pairwise, based on the pairs identified during the matching step [19, 15, 3]. Ensuring efficient performance of the matching step is crucial for the successful completion of the assembly process, particularly in scenarios involving complex fracture patterns [39].

In contrast, the single-step approach directly performs the alignment process [11, 43]. It combines the fragments by incorporating local information from each fragment into the others, allowing the integration of both local and global contextual information. This approach has gained popularity, especially with the advancements in powerful network architectures like transformers [33]. However, it has limitations in handling objects with complex fractures, and it relies on the assumption of prior knowledge that the fragments belong to the same object [43, 32].

3D object matching. As far as we know, previous works in the field of 3D object reassembly have not specifically focused on the matching subtask as a specific problem. Instead, many existing methods incorporate matching as part of their pipeline, with the primary emphasis being on the final reassembly result [44, 10, 40]. While some methods do provide partial analysis of the accuracy in the matching step, these evaluations are often limited in scope and conducted on small, proprietary datasets [39, 34].

The Terracotta Warriors dataset mentioned by Yao *et al*. [39] comprises only 1800 matching pairs and 1560 non-matching pairs, but unfortunately, it is not publicly accessible. Similarly, Wang *et al*. [34] provide a dataset consisting of 4 objects with a total of 12 thin fragments, yet this dataset is also not publicly available. The unavailability of these datasets hampers comprehensive analysis and inhibits meaningful comparisons between previous methods, which limits our understanding of 3D object matching.

3. Background

In this section, we provide a concise background to make it easier to understand our proposal. We formally state the problem we aim to address and describe the concept of graph convolution and its properties.

3.1. Matching problem

Consider a set $S = \{S_1, \cdots, S_n\}$ of multiple point clouds, where each point cloud S_i represents a distinct fragment and n is the total number of point clouds in S. The goal is to determine the matching relationships between pairs of point clouds (S_i, S_j). We define a function $f : S \times S \rightarrow \{0, 1\}$, which takes two point clouds as input and outputs a binary value indicating whether the pair is a match (1) or not (0). The function f is applied to all possible pairs (S_i, S_j), satisfying the condition $i \neq j$. The goal is to find a suitable function f that accurately determines the point cloud pairs in S that should be reassembled together [39].

3.2. Graph convolution

Given a point cloud $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$, where $\mathbf{p}_i \in \mathbb{R}^3$ represents the 3-dimensional coordinates of the i-th point, the graph convolution operation computes the features for each point by aggregating information from its neighboring points. Let $\mathcal{N}_i$ denote the set of neighboring points for point $\mathbf{p}_i$. The graph convolution operation can be expressed as equation 1 where $\mathbf{f}_i$ represents the computed

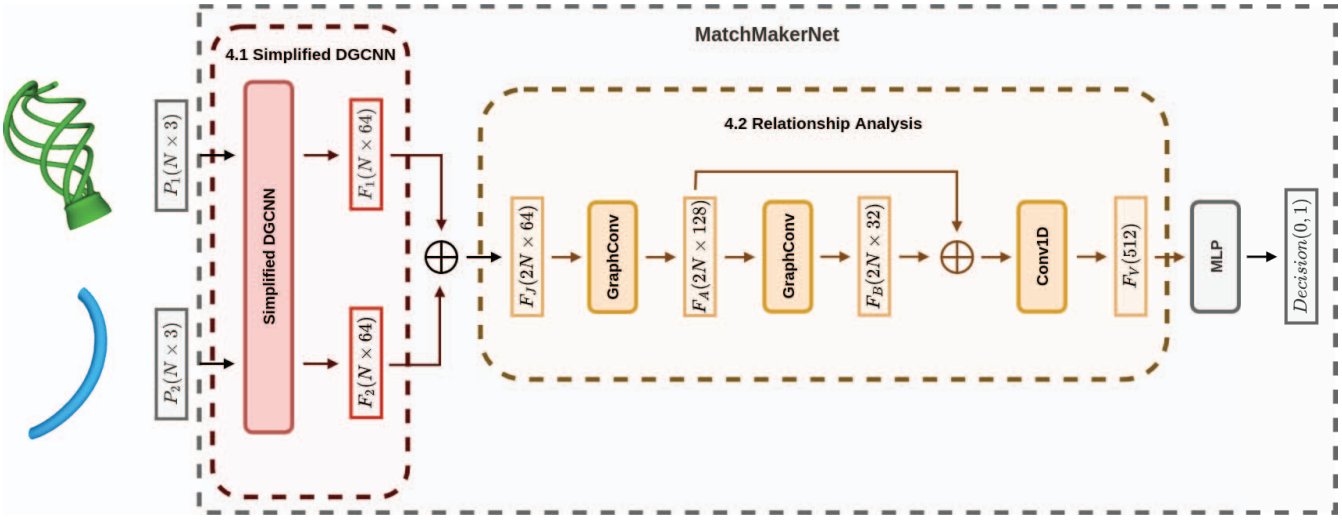


Figure 1: **MatchMakerNet.** Our proposed network architecture is designed for discriminating pairs of fragments that should be matched together. It incorporates our simplified DGCNN (4.1) to extract features from input point clouds. The features are subsequently processed through graph convolutions and combined using a residual connection to perform the relationship analysis (4.2). Following that, an MLP with multiple layers is used to make the final matching decision.

feature for point $\mathbf{p}_i$, $\mathbf{W}$ is a learnable weight matrix, h is an activation function, and $\mathbf{b}$ is a bias term [37].

$$\mathbf{f}_i = h\left(\sum_{\mathbf{p}_j \in \mathcal{N}_i} \mathbf{W}(\mathbf{p}_i - \mathbf{p}_j) + \mathbf{b}\right) \tag{1}$$

By applying the graph convolution iteratively across multiple layers, hierarchical representations of the point cloud can be obtained, allowing for the extraction of abstract and discriminative features that capture local and global patterns and dependencies within the data. This operation exhibits key properties that contribute to its effectiveness. First, it is permutation invariant, guaranteeing that the output features remain unchanged regardless of the order of input points. Additionally, it utilizes local neighborhood information, enabling each point to incorporate features from its surrounding points. Furthermore, the operation is partially invariant to translation, given that it takes into account the local geometry of the patches while retaining global shape information [37].

4. MatchMakerNet

In this section, we introduce MatchMakerNet, a novel network architecture designed to discriminate between pairs of thick fragments that share at least one fractured face. The architecture incorporates graph convolutions as the core component to effectively capture and analyze the relationships between thick fragments, thereby enhancing the precision of the matching process.

MatchMakerNet architecture is shown in Figure 1. First, the shared network extracts features from the input point clouds. To enhance the efficiency of feature extraction, we propose a simplified version of the Dynamic Graph CNN (DGCNN) [37]. This modified approach incorporates the concept of vector neurons, inspired by Deng *et al.* [6] proposal, which exhibit invariance to rotations. By leveraging these vector neurons, MatchMakerNet can extract meaningful features that are robust to rotation transformations, ensuring the network's effectiveness in various scenarios.

Second, the extracted features are concatenated to form a joint feature matrix consisting of 64 features per point. This matrix, denoted as F_J, serves as the input for the subsequent relationship analysis step. This step focuses on assessing the presence of at least one shared fractured face between the fragments. Finally, to make the matching decision, MatchMakerNet employs a multi-layer perceptron (MLP). The MLP consists of concatenated layers with output sizes of 256, 64, and 1, incorporating RELU activation functions in the hidden layers and a sigmoid activation function in the output layer. Through the integration of these steps, MatchMakerNet effectively combines feature extraction, relationship analysis, and decision-making processes.

In Section 4.1, we provide a description of our simplified DGCNN architecture, highlighting the specific adaptations we made to enhance its efficiency. Subsequently, in Section 4.2, we introduce a proposed subnetwork that analyzes the relationship between the fragments.

4.1. Simplified DGCNN

To perform feature extraction for the matching subtask, we employ a simplified version of the DGCNN network,

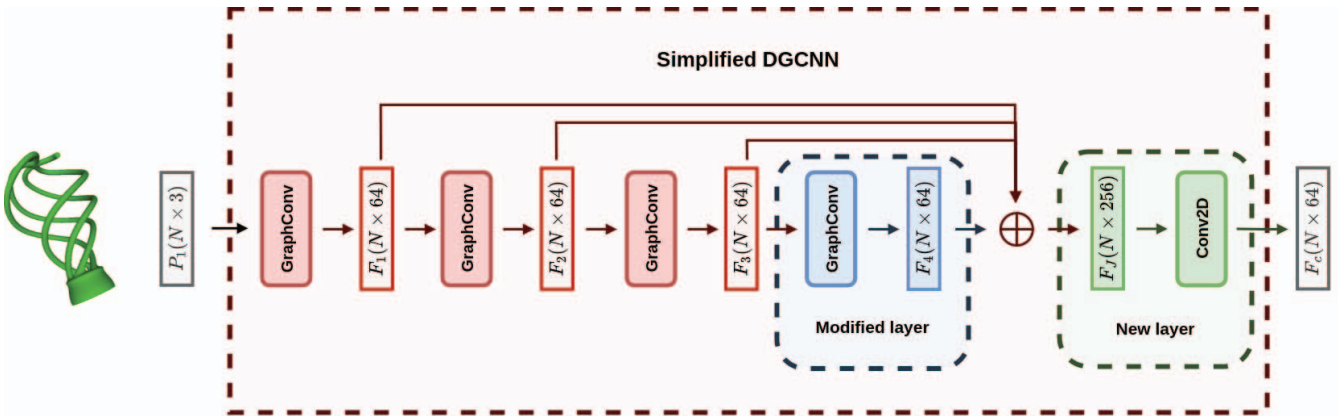


Figure 2: **Simplified DGCNN architecture for feature extraction.** The last graph convolution layer (highlighted in blue) is modified to reduce the number of output channels, while an additional convolution layer (highlighted in green) is added to capture the relationships between the previous layers' representations and further reduce the network.

as illustrated in Figure 2. This customized architecture is specifically designed to achieve two main objectives: reducing its overall size and ensuring invariance to random rotations, enhancing the network's resilience in handling the variations encountered in the matching problem.

The original DGCNN network is composed of four graph convolution layers, with each layer having an output channel size of 64, 64, 64, and 128 respectively. In our approach, we modify the last graph convolutional layer to reduce the number of output channels to 64 instead of 128 (shown in blue in Fig. 2). We also add a convolutional layer with 64 output channels (shown in green in Fig. 2). This supplementary layer facilitates the capture of interlayer relationships within representations while further refining the network's complexity. Finally, we remove the last part of the network, since we do not need to modify the features to fit a classification or segmentation task.

4.2. Relationship Analysis

The relationship analysis subnetwork (shown in yellow in Fig. 1) plays a crucial role in evaluating the relationship between the fragments. By leveraging graph convolution, this subnetwork extracts the features from the joint matrix. This feature matrix undergoes a series of graph convolution layers. Firstly, it passes through a layer with an output of 128 channels, enabling the extraction of rich and informative features. Subsequently, a second graph convolution layer with 64 output channels is applied, further refining the extracted features.

To promote the flow of information and mitigate the issue of diminishing gradients, we adopt a strategy involving the concatenation of outputs from the first and second graph convolution layers. This operation yields a composite matrix, denoted as F_J, which establishes a residual connection, thereby ensuring the joint influence of both representation sets on the output. Following this, a 1D convolutional generates the feature vector F_V, signifying the interaction among fragments. Through the amalgamation of insights garnered from diverse convolutions, the subnetwork captures and assimilates pertinent attributes for evaluating inter-fragment relationships.

5. Experiments

In this section, we evaluate the effectiveness of our proposed feature extractor and matching network, we conduct a set of experiments on various scenarios and datasets. First, we compare our simplified DGCNN with both the original DGCNN version and other state-of-the-art methods in Section 5.1. Specifically, we assess the performance of the methods on 3D objects augmented with random rotations to evaluate their robustness in handling the matching subtask under real-world scenarios where fragment orientations lack a canonical direction. Subsequently, in Section 5.2, we delve into the analysis and evaluation of our MatchMakerNet, with a particular focus on its performance in both real-world and synthetic fractured objects. These experiments allow us to assess the generalization and versatility of our proposed method.

Datasets. Both synthetic and real-world datasets are available for working on 3D object reassembly. Real-world datasets often involve scanned fragmented objects, but can be limited in size and ground truth information. In our experiments, we utilize three datasets: ModelNet40 [41], comprising 12,311 CAD models from 40 categories; Breaking Bad [29], with 10,000 synthetic fractured objects, including cultural artifacts and everyday items; and Puzzles 3D [12], consists of 6 real-world fragmented cultural her-

Method	Overall Accuracy	Class Accuracy	Inference Time (ms)	Model Size (MB)	Parameters (M)
PointNet [25]	50.32 %	41.68 %	6.65	6.28	0.82
PointNet++ [26]	86.18 %	81.71 %	39.11	11.32	1.48
DGCNN [37]	88.57 %	84.27 %	8.94	13.85	1.81
VN-PointNet [6]	87.40 %	83.53 %	9.03	15.12	1.97
VN-DGCNN [6]	90.56 %	**87.51** %	18.08	22.17	2.90
Ours	89.91 %	83.43 %	**4.58**	**1.63**	**0.21**
VN-Ours	**91.02** %	87.07 %	6.41	2.55	0.33

Table 1: **Classification on ModelNet40**. Performance comparison of different methods on various evaluation metrics, including overall accuracy, class accuracy, inference time, model size, and number of parameters. Methods evaluated include PointNet [25], PointNet++ [26], DGCNN [37], VN-PointNet [6], VN-DGCNN [6], our proposed method, and VN-Ours.

itage artifacts, providing a representative sample of practical fracture scenarios with 81 total fragments.

5.1. Simplified DGCNN

Experimental Setup. We train two versions of our simplified DGCNN, one based on the original DGCNN and the other based on VN-DGCNN, which utilizes vector neurons for rotation invariance. The training and validation are performed on the train and test partition of the ModelNet40 dataset [42] using the cross-entropy loss function over 400 epochs. For evaluation, we utilize the weights of the model that achieved the highest accuracy during training, and assess its performance on objects with random rotations. The evaluation metrics includes overall accuracy, class accuracy, inference time, model size, and the number of parameters. To ensure consistent evaluation, we employ the default train and test split defined in the ModelNet40 dataset.

Baseline Methods. In order to assess the performance of our approach, we compare it against several state-of-the-art methods, including PointNet [25], PointNet++ [26], DGCNN [37], VN-PointNet [6], and VN-DGCNN [6]. We retrain each method using the publicly available code from the original papers and evaluate all models under similar point cloud transformations as our proposed model.

Results. Table 1 compares the performance of various methods based on evaluation metrics. PointNet achieves an overall accuracy of 50.32% and class accuracy of 41.68%, while PointNet++ improves the results with an overall accuracy of 86.18% and class accuracy of 81.71%. DGCNN achieves further enhancement with an overall accuracy of 88.57% and class accuracy of 84.27%. VN-PointNet and VN-DGCNN, utilizing the vector neuron approach, achieve overall accuracies of 87.40% and 90.56%, and class accuracies of 83.53% and 87.51%, respectively.

Our method surpasses these results, achieving an overall accuracy of 89.91% and class accuracy of 83.43%, with improved efficiency in terms of inference time (3.56 ms), model size (1.63 MB), and parameters (0.21 million). Incorporating the vector neuron approach in our method (VN-Ours) further improves the overall accuracy to 91.02% and class accuracy to 87.07%, with slightly increased inference time (6.41 ms), model size (2.55 MB), and parameters (0.33 million).

The results presented in Table 1 demonstrate the effectiveness of our method with the vector neuron approach in maintaining the accuracy and maintaining the efficiency compared to existing state-of-the-art methods. On one hand, our simplified version of DGCNN compared with the original DGCNN offers significant improvements in efficiency, reducing the inference time by 1.95 times and memory usage by 8.50 times. Although there is a slight loss of 0.87% in class accuracy, it compensates with a gain of 1.34% in overall accuracy. On the other hand, our simplified DGCNN with the vector neuron approach compared with VN-DGCNN provides even greater efficiency, reducing the inference time by 2.82 times and memory usage by 8.69 times. It experiences a slight decrease of 0.44% in class accuracy, but achieves a commendable 0.46% increase in overall accuracy.

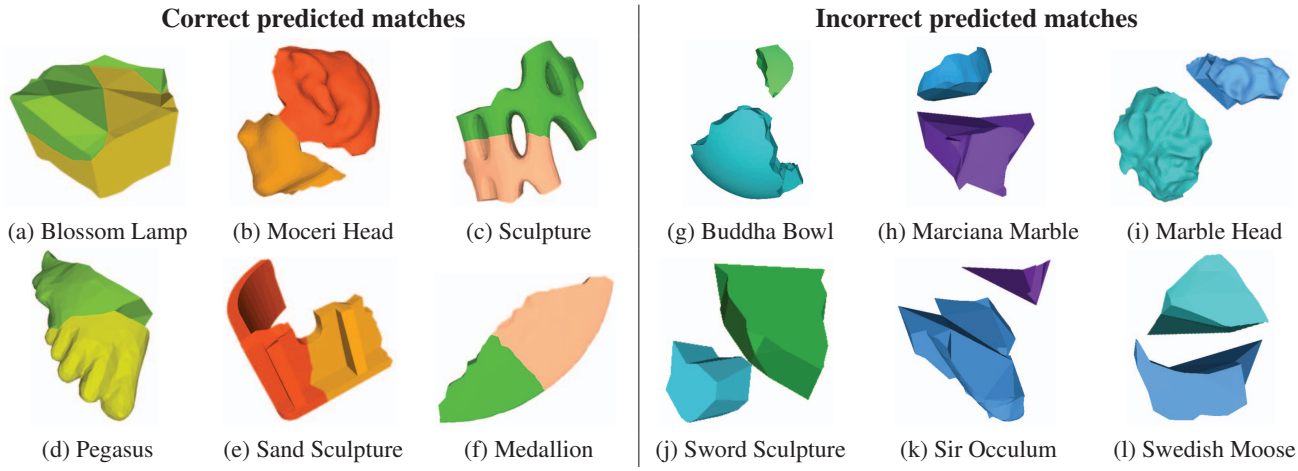
Considering these results, the choice of method depends on the specific requirements of the task at hand. If efficiency is paramount, our simplified DGCNN is the preferred option. For a balance between high overall accuracy and efficiency, our simplified DGCNN with vector neurons (Ours-VN) stands out as the optimal choice. However, if preserving the best possible class accuracy is crucial, VN-DGCNN should be considered. In our case, where both speed and accuracy are important, we have chosen to utilize Ours-VN as the feature extractor for MatchMakerNet.

5.2. MatchMakerNet

Experimental Setup. We train MatchMakerNet on the Artifact subset of Breaking Bad [29]. We utilized the binary cross-entropy (BCE) loss function and implemented 5-fold cross-validation for training. Each fold undergo a maximum of 100 epochs, with Early Stopping to stop the training if the validation loss failed to decrease for 20 consecutive epochs. In addition, we balanced the train subset to have 70% positive examples and 30% negative examples. This improved pattern identification and classification, given that the origi-

	Loss		Accuracy		F1 score	
Fold	**70/30**	**all/all**	**70/30**	**all/all**	**70/30**	**all/all**
1	0.2109	0.2475	86.31 %	87.55 %	86.71 %	89.78 %
2	0.2243	0.2240	81.73 %	88.21 %	83.74 %	90.35 %
3	0.2359	0.2398	80.53 %	86.33 %	81.24 %	88.61 %
4	0.2140	0.2268	83.85 %	87.34 %	85.23 %	90.41 %
5	0.2184	0.2132	81.90 %	87.12%	84.12 %	91.06 %
Mean	0.2207	0.2303	82.86 %	87.31 %	84.21 %	90.04 %
Std	0.0088	0.0121	2.05 %	0.61 %	1.81 %	0.82 %

Table 2: **Evaluation on Artifact (Breaking Bad).** We show the performance metrics (loss, accuracy, and F1 score) of the model trained and evaluated with different distributions and folds. The model is trained with a distribution of 70% positive and 30% negative examples, and evaluated using both the 70/30 and original distributions. The evaluation is conducted over 5 folds, and the reported mean and standard deviation of the metrics demonstrate the consistency of the model.



(a) Blossom Lamp (b) Moceri Head (c) Sculpture (g) Buddha Bowl (h) Marciana Marble (i) Marble Head

(d) Pegasus (e) Sand Sculpture (f) Medallion (j) Sword Sculpture (k) Sir Occulum (l) Swedish Moose

Figure 3: **Examples of pairs found by the network on the Artifact subset**. On the left, we present six correct predicted matches, while on the right, we showcase size incorrect predicted matches based on the ground truth labels.

nal imbalanced dataset approximately has 10% positive examples and 90% negative examples, with the positive instances posing a greater challenge.

Following the training of each fold, we evaluated the performance on the test subset, employing two different distributions. The first evaluation used a dataset with a distribution of 70% positive examples and 30% negative examples, while the second evaluation employed the original dataset distribution. Finally, to assess the model's generalization capabilities, we conducted evaluations on additional datasets, including the Everyday subset of Breaking Bad and Puzzles 3D. We measured the model's performance using metrics such as loss, accuracy and F1 score.

To determine the ground truth for the matching task, we compare each fragment $\mathcal{P}_t$ to the other fragments of the same object $\mathcal{P}_i \in S$, excluding self-comparisons ($i \neq t$). For each comparison, we calculate the minimum distance between the points in $\mathcal{P}_t$ and $\mathcal{P}_i$. If the minimum distance is less than a threshold η and there exists at least one common face between the selected vertices, we assign a label of 1 to indicate a match; otherwise, a label of 0 is assigned. The common face check ensures that there is at least one shared face between the selected vertices, indicating a meaningful match.

Results. Table 2 presents the loss, accuracy and F1 score on the evaluation set for models trained in each fold. According to the results in this table, a superior performance is observed in the original distribution. In this distribution, the mean accuracy reaches approximately 87%, while the mean F1 score approaches 90%, surpassing the performance of the 70/30 distribution, which achieves a mean accuracy of 83% and a mean F1 score of 84%. This disparity can be attributed to the prevalence of negative examples in the original distribution, which are relatively easier to identify compared to positive examples. This observation proves advantageous in the real-world scenario of all-to-all comparison between the available fragments.

Moreover, the original distribution shows a slightly

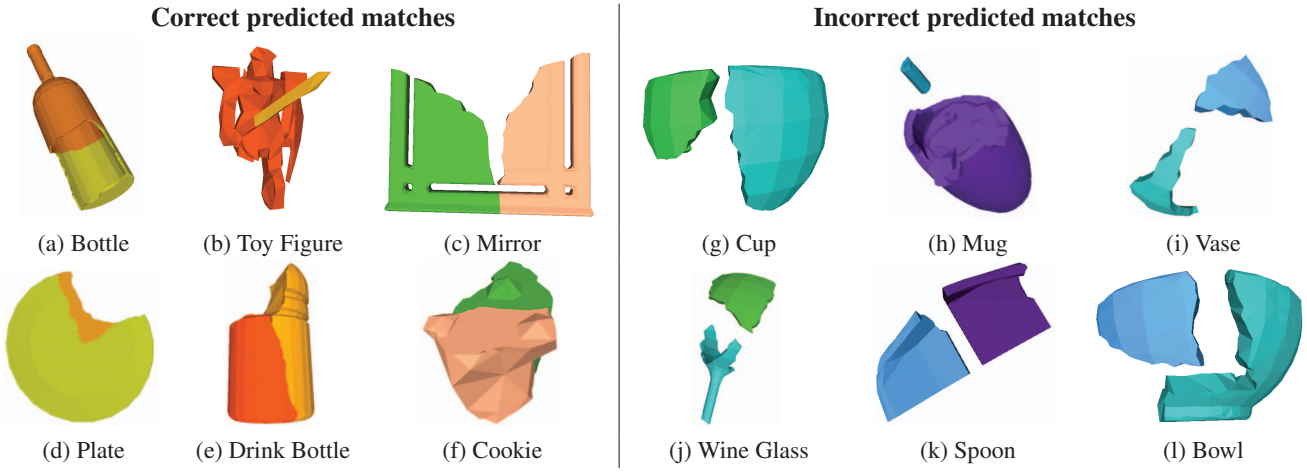


Figure 4: **Examples of pairs found by the network on the Everyday subset**. On the left, we present six correct examples, while on the right, we showcase six incorrect examples based on the ground truth labels.

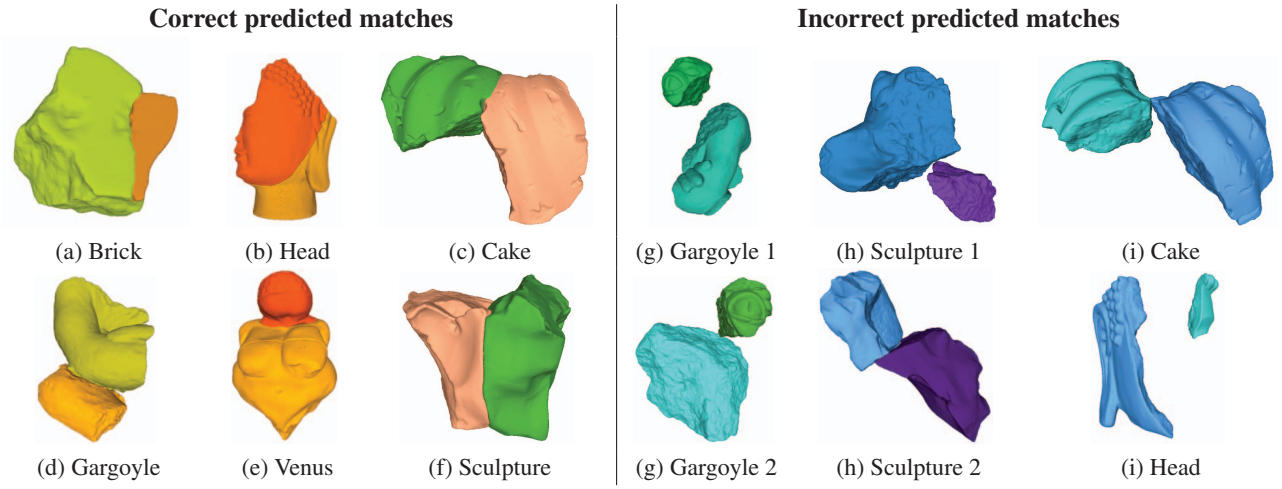


Figure 5: **Examples of pairs found by the network on the Puzzles 3D dataset**. On the left, we present six correct examples, while on the right, we showcase six incorrect examples based on the ground truth labels.

higher loss compared to 70/30 distributions. However, considering the observed standard deviation, this difference is not substantial. Additionally, the results show low variation between folds, indicating that the model is stable and produces consistent results when it is trained with different training and validation sets.

Figure 3 showcases four examples of fragment pairs that should be reassembled based on the results obtained from the matching network. The left examples (Fig. 3a to 3f) are correct matches, they highlight the network's capability to identify pairs with both complete and partially fractured faces. This ability is particularly relevant in real-life scenarios where complete face sharing is less common and considered an idealized scenario. By successfully identifying matches with partial fractured faces, the network demonstrates its practical applicability in handling realistic reassembly tasks.

In the right examples (Fig. 3g to 3i), where the network's outcomes were incorrect, two discernible issues come to light. First, the network tends to experience confusion when confronted with similar fracture patterns in fragments that should remain separate. Second, certain fracture patterns lack complex details, particularly when the fractured faces appear nearly flat. These observations indicate potential challenges to the network in accurately discerning subtle differences and capturing fine-grained information from certain types of fractures.

Table 3 presents the results obtained by the matching network trained on Artifact subset and evaluated on the Everyday subset and Puzzles 3D dataset. According to the ta-

ble, the network demonstrates the ability to generalize to 3D objects from other datasets without significant loss in its performance. Furthermore, the Puzzles 3D dataset poses a greater challenge for the network compared to the Everyday subset. MatchMakerNet achieves an accuracy of 87% and an F1 score of 89% on Everyday, whereas it achieves an accuracy of 83% and an F1 score of 85% on Puzzles 3D. This difference can be attributed to the fact that both Artifact and Everyday are subsets of the synthetic dataset Breaking Bad, whereas Puzzles 3D has more complex real-world fracture patterns.

Test Dataset	Loss	Accuracy	F1 score
Everyday	0.3204	86.93 %	89.27 %
Puzzles 3D	0.3599	83.03 %	85.24 %

Table 3: **Comparison of results on Everyday and Puzzles 3D**. The table presents the loss, accuracy, and F1 score for the Everyday and Puzzles 3D test datasets.

In Figure 4, we observe examples of pairs on Everyday subset that need to be reassembled based on the results obtained by MatchMakerNet. The left side (Fig. 4a to 4f) showcases correct examples, where the network successfully solves matching for thin fragments, despite being primarily trained on thick fragments. Additionally, the network exhibits the ability to identify matching pairs in fragments of different sizes, as seen in the case of the Toy Figure, where the sword is much smaller than the robot.

Conversely, the incorrect examples exhibit patterns similar to those identified in the Artifact subset. The network misclassifies fragments with similar fracture patterns as positive, even though they should not be reassembled together. Furthermore, some fractures lack detail and are almost flat, making it challenging to accurately identify fracture details and choose the correct pair. It is important to note that some of these limitations may be related to the fact that we are using synthetically generated fragments.

Figure 5 presents examples of fragment pairs from the Puzzles 3D dataset that MatchMakerNet identifies as candidates for reassembly. On the left side (Fig. 5a to 5f), correct examples are shown, while on the right side (Fig. 5g to 5i), incorrect examples are displayed. In the correct examples, we notice that fragment pieces can vary in size and still achieve a correct matching prediction. Additionally, the network can generalize correspondence for thin fragments. Overall, we have reached similar observations as those obtained with the Artifact and Everyday subsets.

Similarly, the analysis of incorrect examples reveals that the network often struggles to differentiate fragment pairs with similar fracture patterns. In the specific case of Gargoyle, although there are notable similarities in the fracture patterns, subtle differences also exist. Possible factors contributing to these prediction errors include preprocessing steps that might result in the loss of certain distinguishing features and the use of synthetic data for training, which lacks the same level of detail as real fragments. Furthermore, it is important to highlight that some errors occur because the network identifies fragments that should be joined if they share any common point or line, without taking into account the presence of at least one fractured face.

6. Conclusion

In this paper, we propose a simplified DGCNN as a feature extractor and MatchMakerNet as a network to solve the matching subtask in the reassembly problem. Both approaches leverage the power of graph convolution and its properties to effectively represent local and global features. The results demonstrate that our simplified DGCNN is capable of maintaining the performance of the original networks with minimal loss, while significantly reducing memory and time costs.

Additionally, we evaluated the performance of MatchMakerNet after training it on the Artifact subset. The network achieved an accuracy of 87.31% on the original distribution. However, when assessing its generalization capabilities, we observed that it encounters more challenges when handling the Puzzles 3D subset, which consists of real-world fractures. These fractures pose greater difficulty due to their diverse and complex nature, compared to the relatively less detailed fracture patterns found in the Breaking Bad dataset.

These findings highlight the potential of MatchMakerNet and the simplified DGCNN in automating the matching step of fragmented objects, particularly in the context of cultural heritage analysis. This ability is of great importance in preserving and understanding cultural artifacts, as it enables researchers and archaeologists to gain insights into the original form and structure of these objects.

7. Future Work

In our future work, we plan to further enhance the pipeline by incorporating the alignment step, which will enable us to perform end-to-end experiments and achieve complete object reassembly. For the simplified DGCNN, we aim to evaluate its performance on a wider range of datasets and tasks beyond ModelNet40, including segmentation and recognition. In the case of MatchMakerNet, our future work will be focused on improving its performance on real-world fragments. To achieve this, we intend to explore the incorporation of more detailed fracture faces, especially for fragments with nearly flat fractures. These future research directions will contribute to the overall advancement and applicability of our methods in practical scenarios.

References

[1] Noémie Bonneau, July Bouhallier, Caroline Simonis, Michel Baylac, Olivier Gagey, and Christine Tardieu. Technical note: Shape variability induced by reassembly of human pelvic bones. *American Journal of Physical Anthropology*, 148(1):139–147, 2012.

[2] Omer Cakir and Vasif Nabivev. A region alignment and matching method for fractured object reassembly. In *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pages 528–532, 2021.

[3] Yun-Chun Chen, Haoda Li, Dylan Turpin, Alec Jacobson, and Animesh Garg. Neural shape mating: Self-supervised object assembly with adversarial shape priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[4] Yan Cui, Sebastian Schuon, Sebastian Thrun, Didier Stricker, and Christian Theobalt. Algorithms for 3d shape scanning with a depth camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1039–1050, 2013.

[5] Roberto de Lima-Hernandez and Maarten Vergauwen. A hybrid approach to reassemble ancient decorated block fragments through a 3d puzzling engine. *Remote Sensing*, 12(16), 2020.

[6] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas. Vector neurons: a general framework for so(3)-equivariant networks. *arXiv preprint arXiv:2104.12229*, 2021.

[7] Stuart Eve. Losing our senses, an exploration of 3d object scanning. *Open Archaeology*, 4(1):114–122, 2018.

[8] Abid Haleem, Mohd Javaid, Ravi Pratap Singh, Shanay Rab, Rajiv Suman, Lalit Kumar, and Ibrahim Haleem Khan. Exploring the potential of 3d scanning in industry 4.0: An overview. *International Journal of Cognitive Computing in Engineering*, 3:161–171, 2022.

[9] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1–11, October 2021.

[10] Je Hyeong Hong, Seong Jong Yoo, Muhammad Arshad Zeeshan, Young Min Kim, and Jinwook Kim. Structure-from-sherds: Incremental 3d reassembly of axially symmetric pots from unordered and mixed fragment collections. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5423–5431, 2021.

[11] Jialei Huang, Guanqi Zhan, Qingnan Fan, Kaichun Mo, Lin Shao, Baoquan Chen, Leonidas Guibas, and Hao Dong. Generative 3d part assembly via dynamic graph learning. In *The IEEE Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[12] Qi-Xing Huang, Simon Flöry, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graph.*, 25(3):569–578, 2006.

[13] Sheng hui Liao, Chao Xiong, Shu Liu, Ying qi Zhang, and Chun lin Peng. 3d object reassembly using region-pair-relation and balanced cluster tree. *Computer Methods and Programs in Biomedicine*, 197:105756, 2020.

[14] Irwansyah, D.B. Redyarsa, Jiing-Yih Lai, Terence Essomba, and Pei-Yuan Lee. Detecting and removing overlap meshes for the assembly of 3d-printed fractured bones. In *2018 IEEE International Conference on Applied System Invention (ICASI)*, pages 362–365, 2018.

[15] Caiqin Jia, Ligang He, Xiaowen Yang, Xingcheng Han, Bobo Chang, and Xie Han. Developing a reassembling algorithm for broken objects. *IEEE Access*, 8:220320–220334, 2020.

[16] Jr. Key, Marcus M., Patrick N. Wyse Jackson, and Stephen H. Felton. Intracolony variation in colony morphology in reassembled fossil ramose stenolaemate bryozoans from the Upper Ordovician (Katian) of the Cincinnati Arch region, USA. *Journal of Paleontology*, 90(3):400–412, 2016.

[17] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: Differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[18] Jingtao Li, Jian Zhou, Yan Xiong, Xing Chen, and Chaitali Chakrabarti. An adjustable farthest point sampling method for approximately-sorted point cloud data, 2022.

[19] Qunhui Li, Guohua Geng, and Mingquan Zhou. Pairwise matching for 3d fragment reassembly based on boundary curves and concave-convex patches. *IEEE Access*, 8:6153–6161, 2020.

[20] Bin Liu, Mingzhe Wang, Xiaolei Niu, Shengfa Wang, Song Zhang, and Jianxin Zhang. A fragment fracture surface segmentation method based on learning of local geometric features on margins used for automatic utensil reassembly. *Computer-Aided Design*, 132:102963, 2021.

[21] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11596–11603, 2020.

[22] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022.

[23] Ehsan Nezhadarya, Ehsan Taghavi, Ryan Razani, Bingbing Liu, and Jun Luo. Adaptive hierarchical down-sampling for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[24] Georgios Papaioannou, Tobias Schreck, Anthousis Andreadis, Pavlos Mavridis, Robert Gregor, Ivan Sipiran, and Konstantinos Vardis. From reassembly to object completion: A complete systems pipeline. *J. Comput. Cult. Herit.*, 10(2), 2017.

[25] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[26] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[27] Shi Qiu, Saeed Anwar, and Nick Barnes. Geometric back-projection network for point cloud classification. *IEEE Transactions on Multimedia*, 24:1943–1955, 2022.

[28] Mhairi Reid, Emese M Bordy, Wendy L Taylor, Stephan G le Roux, and Anton duPlessis. A micro X-ray computed tomography dataset of fossil echinoderms in an ancient obrution bed: a robust method for taphonomic and palaeoecologic analyses. *GigaScience*, 8(3), 2018.

[29] Silvia Sellán, Yun-Chun Chen, Ziyi Wu, Animesh Garg, and Alec Jacobson. Breaking bad: A dataset for geometric fracture and reassembly. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[30] Tae-Geun Son, Jusung Lee, Jeonghun Lim, and Kunwoo Lee. Reassembly of fractured objects using surface signature. *Vis. Comput.*, 34(10):1371–1381, 2018.

[31] Petra Urbanová, Ann H. Ross, Mikoláš Jurda, and Ivana Šplíchalová. The virtual approach to the assessment of skeletal injuries in human skeletal remains of forensic importance. *Journal of Forensic and Legal Medicine*, 49:59–75, 2017.

[32] Lev Utkin, Maxim Kovalev, and Ernest Kasimov. An explanation method for siamese neural networks. In Nikita Voinov, Tobias Schreck, and Sanowar Khan, editors, *Proceedings of International Scientific Conference on Telecommunications, Computing and Control*, pages 219–230, 2021.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[34] Haiping Wang, Yufu Zang, Fuxun Liang, Zhen Dong, Hongchao Fan, and Bisheng Yang. A probabilistic method for fractured cultural relics automatic reassembly. *J. Comput. Cult. Herit.*, 14(1), 2021.

[35] Lei Wang, Junjun Pan, and Qiangqiang Yao. Virtual reassembly of fractured bones for orthopedic surgery. In *2018 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 20–27, 2018.

[36] Piao Wang, Guohua Geng, Xiaofeng Wang, and Yi Wang. Method for splicing fragments based on constructing surface texture and fracture boundary tuple. 2018.

[37] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[38] Cheng Wen, Baosheng Yu, and Dacheng Tao. Learnable skeleton-aware 3d point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17671–17681, June 2023.

[39] Wenmin Yao, Tong Chu, Wenlong Tang, Jingyu Wang, Xin Cao, Fengjun Zhao, Kang Li, Guohua Geng, and Mingquan Zhou. Sppd: A novel reassembly method for 3d terracotta warrior fragments based on fracture surface information. *ISPRS International Journal of Geo-Information*, 10(8), 2021.

[40] Congli Yin, Mingquan Zhou, Yachun Fan, and Wuyang Shui. Template-guided 3d fragment reassembly using gds. In Yongtian Wang, Zhiguo Jiang, and Yuxin Peng, editors, *Image and Graphics Technologies and Applications*, pages 432–441, 2018.

[41] A. Khosla F. Yu L. Zhang X. Tang J. Xiao Z. Wu, S. Song. 3d shapenets: A deep representation for volumetric shapes. In *Computer Vision and Pattern Recognition*, 2015.

[42] A. Khosla F. Yu L. Zhang X. Tang J. Xiao Z. Wu, S. Song. 3d shapenets: A deep representation for volumetric shapes. In *Computer Vision and Pattern Recognition*, 2015.

[43] Rufeng Zhang, Tao Kong, Weihao Wang, Xuan Han, and Mingyu You. 3d part assembly generation with instance encoded transformer. *IEEE Robotics and Automation Letters*, 7:9051–9058, 2022.

[44] Yuhe Zhang, Kang Li, Xiaoxue Chen, Shunli Zhang, and Guohua Geng. A multi feature fusion method for reassembly of 3d cultural heritage artifacts. *Journal of Cultural Heritage*, 33:191–200, 2018.