Towards Pointsets Representation Learning via Self-Supervised Learning and Set Augmentation

Pattaramanee Arsomngern, Cheng Long, Supasorn Suwajanakorn, and Sarana Nutanong

Abstract—Deep metric learning is a supervised learning paradigm to construct a meaningful vector space to represent complex objects. A successful application of deep metric learning to pointsets means that we can avoid expensive retrieval operations on objects such as documents and can significantly facilitate many machine learning and data mining tasks involving pointsets. We propose a self-supervised deep metric learning solution for pointsets. The novelty of our proposed solution lies in a self-supervision mechanism that makes use of a distribution distance for set ranking called the Earth's Mover Distance (EMD) to generate pseudo labels and a pointset augmentation method for supporting the learning solution. Our experimental studies on documents, graphs, and point clouds datasets show that our proposed solutions outperform baselines and state-of-the-art approaches under the unsupervised settings. The learned self-supervised representation can also be used as a pre-trained model, which can boost downstream tasks with a fine-tuning step and outperform state-of-the-art language models.

Index Terms—set retrieval, deep metric learning, self-supervised learning, triplet loss, earth mover's distance

1 Introduction

Distance measuring has been used to find the similarity between the query data point and other samples in the dataset. The calculated distance can be applied in various machine learning algorithms to make predictions. However, pointsets, which have been widely used for representing documents [1], graphs (as a set of vertex embeddings) [2], and 3D models (point clouds) [3], require a complex distance function to compute a distance between each sample. In particular, one of the effective distribution distances for pointsets called Earth Mover's Distance (EMD) [4] requires $O(n^3 \log n)$ per one comparison. Deep metric learning can alleviate this high computation cost problem by encoding raw pointsets into new representations in a Euclidean space with a lower distance measuring cost. Several studies proved that deep metric learning approaches achieve state-of-the-art results in various tasks, including image retrieval [5], [6], face recognition [7], [8], [9], person re-identification [10], [11], trajectory similarity search [12], and pointsets similarity search [13] in terms of classification accuracy and computation cost. However, these techniques mostly require supervision, which cannot be applied to unlabeled data.

Apart from learning representations from labels, self-supervised techniques for embedding data points have recently received growing research attention. These techniques focus on mapping unlabeled images into new representations by learning similarities between augmented views of images (i.e., contrastive learning) [14], [15], [16], [17], [18]. The main challenge to apply self-supervised learning techniques to pointset is the set's permutation invariant property. This constraint precludes generic augmentation techniques such as linear interpolation for applying it with pointsets directly. On the other hand, some studies try to learn pointsets in an unsupervised way using the auto-encoder [19], [20]. However, its representation still has poorer performance than representation learned from techniques using labels (e.g., deep metric learning [13]).

In this paper, we propose a new deep metric learning method to learn representations for pointsets in a self-supervised way. Our proposed method includes a new loss and an augmentation technique for pointsets. For ease of understanding, we call our framework *Pointset2Vec*. The proposed method can encode any unlabeled pointsets into a new embedding space that provides the simplicity of distance calculation and achieves state-of-the-art performance for pointset representation learning.

Our new loss function utilizes EMD to infer a pseudo-label for each mini-batch. Instead of training by the deep metric learning-based technique with triplets constructed based on the ground-truth labels, which are hard to acquire, we use triplets generated from pseudo-labels to learn a new embedding space. Moreover, our loss function is also compatible with any distance function other than EMD. We also introduce *PointSwap*, a new augmentation technique that randomly swaps elements of two similar sets to generate a new set. PointSwap enlarges our training set and extends our loss function to learn the similarity between the original and augmented pointsets, leading to better representation performance. Furthermore, we also provide a comprehensive study of transfer learning strategies to transfer the representation learned by our loss to a target problem.

We evaluate our proposed method with documents (set of word vectors) [1], graphs [21], and point clouds [22] datasets. The combination of our loss function and PointSwap achieves state-of-the-art performance in docu-

- *P. Arsomngern, S. Suwajanakorn, and S. Nutanong are with School of Information Science and Technology, Vidyasirimedhi Institute of Science and Technology, Thailand. E-mail: {pattaramanee.a_s19, supasorn.s, snutanon}@vistec.ac.th*
- *C. Long is with School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail:c.long@ntu.edu.sg*

Manuscript received XXXXX; revised XXXXX.

ment classification tasks and has comparable performance in graph and point cloud classification tasks under the setting of training with unlabeled datasets. Our representation also has comparable performance with large and specific natural language models, i.e., RoBERTa [23], when fine-tuned with fully labeled datasets and partially labeled datasets (i.e., semi-supervised settings). Furthermore, we show an adaptation of our learned representation to unsupervised tasks (i.e., distance approximation and density estimation) and provide an ablation study of PointSwap to investigate its effectiveness.

Based on our conference version [13], this extended work improves the previous work's performance and further demonstrates its generalizability. We demonstrate the overall contributions of our work along with the newly introduced elements in this version as follows:

1) *Self-supervised loss function:* We propose a novel self-supervised deep metric learning loss function based on EMD, or any suitable distance. The model learned using our loss function can map an unlabeled pointset to a new representation, which helps avoid the expensive computational cost of distance calculation.
   We further propose an augmentation-based self-supervised loss, which is inspired by recent studies of contrastive learning in computer vision. This augmentation-based loss is newly introduced in this extended version. The new form of loss serves the model for learning a representation by maximizing the similarity between the original and augmented documents.
2) *PointSwap:* We find that the representation performance of our previous work, which adopts a self-supervised triplet loss with triplets generated from EMD values for learning, could be limited by EMD retrieval performance on the raw pointsets. We tackle this issue by introducing augmented pointsets called *PointSwap* to generate triplets based on augmented views concurrently with the EMD-based triplets. This PointSwap technique is newly introduced in this extended version.
3) *Transfer learning strategies:* We introduce transfer learning strategies to improve the learning performance in downstream tasks for pointsets by fine-tuning with fully-labeled and partially labeled datasets.
4) *Extensive experiments:* We conducted extensive experiments to compare our proposed solution against state-of-the-art techniques for pointsets in both unsupervised and supervised settings.
   In particular, the following experiments are newly included in this extended version. We compare our method to state-of-the-art representation learning in computer vision (i.e., SimCLR [17]) by implementing a contrastive learning pipeline for pointsets. Compared to the previous work that has only been verified on documents, we added experiments on graph datasets, a point cloud dataset, and a study on applying chamfer distance for mining triplets in our loss function. We additionally detail the time complexity of each competitor in runtime analysis and provide a more comprehensive runtime comparison between our method and graph neural network [24]. We also conduct ablation studies on augmentation choices to investigate the impact of each augmentation on self-supervised pointset representation learning. A study of transferring the learned representation to an unsupervised task is also added to present the capability of our pre-trained model on different downstream tasks.
5) *New findings and performance improvement (in the extended version):* PointSwap improves the classification accuracy of our conference version and becomes a new state-of-the-art unsupervised and semi-supervised document classification model by gaining 0.91% and 2.23% in BBCSports dataset and 0.20% and 1.33% in Amazon dataset, respectively.

The rest of this paper is organized as follows. Section 2 presents related techniques. The problem formulation and the proposed solution are discussed in Sections 3 and 4, respectively, with the additional application discussed in Section 5. We then report the experimental setup and the results in Sections 6 and 7, respectively. Finally, Section 8 concludes the presentation.

2 Related Work

2.1 Pointsets Similarity Search

Pointset is a versatile format to represent unstructured data such as documents [1], video clips [25], and point clouds [26]. Various distribution distances have been studied for applying in pointsets similarity search, e.g., Earth Mover's Distance (EMD) [4], Hausdorff Distance, and Chamfer Distance (CD) [27]. The main challenge of finding similar pointsets lies in the cost of distance calculation. For example, using the EMD, which is a well-known measure for pointsets, we end up with a cubic (or cubic-logarithmic) complexity with respect to the number of elements per distance calculation.

Due to the stated challenge, researchers have been dedicating attention to improving the efficiency of set similarity retrieval. Traditional approaches include (i) distance approximation [28], [29], [30]; (ii) pruning [31]. While these techniques provide an improvement over the original EMD baseline, they still suffer from issues of computational cost. Furthermore, distance functions could not be fine-tuned on the downstream task, e.g., a classification problem. In the next subsection, we will discuss how the concept of deep metric learning can address these issues.

2.2 Supervised Deep Metric Learning

Distance-based deep metric learning is a popular technique to learn an embedding function in a supervised way, such that the distance between any two samples in the same class (i.e., positives) are significantly nearer than those between samples from other classes (i.e., negatives). There are two main methods for distance-based learning: *pair-based* [32] and *triplet-based* [33]. Let $\left\|f(x^a) - f(x^p)\right\|$ and $\left\|f(x^a) - f(x^n)\right\|$ be the distance from one sample acting as an anchor x^a to the positive sample x^p and the distance from

x^a to the negative sample x^n, respectively. And let α_p and α_n be the positive and negative margins. For the pair-based method, we try to keep $\|f(x^a) - f(x^p)\|$ smaller than α_p and $\|f(x^a) - f(x^n)\|$ greater than α_n. For the triplet-based method, we also use positive and negative pairs. We try to keep $\|f(x^a) - f(x^n)\|$ greater than $\|f(x^a) - f(x^p)\|$ by a given margin α. Results from various experiments [5], [7], [10], [11], [34] show that using a shared reference point in the triplet-based method provides an improvement over the pair-based one.

Various studies demonstrate the importance of triplet sampling. Schroff et al. [7] show that focusing on critical negatives, i.e., those easy to be mistaken, provides a performance improvement over selecting negatives randomly. Other investigations on triplet mining include techniques that make use of the same anchor and positive instances with different negative instances [5], [11], [34], [35]. In our work, we employ the triplet loss with a semi-hard negative sampling [7] and propose a self-supervised method to train it without labels, which were traditionally required.

2.3 Unsupervised and Self-supervised Representation Learning

Recently, unsupervised and self-supervised representation learning has received growing research attention, especially in the field of computer vision. Dosovitskiy et al. [14] formulate a self-prediction problem in which each instance is its own distinct class by using *contrastive loss*. Wu et al. [36] introduce a non-parametric network, which has an improvement over the method from Dosovitskiy et al. [14]. Ye et al. [15] propose a self-supervised method that includes data augmentation into the training process and maximizing similarity between augmented images and original images. However, these methods still perform worse than supervised pre-training method using ImageNet labels [15].

He et al. [16] introduce a new learning method that combines data augmentation [15] and non-parametric network [36] into the training process called MoCo. This method can surpass the supervised pre-trained model on downstream tasks. On the other hand, Chen et al. [17] discard complex non-parametric updates in MoCo [16] and proposed a simpler training pipeline using two different augmented views of the original images instead of the original-augmented pair called SimCLR. Their study also reveals that stronger augmentation and larger batch sizes can increase the representation performance in transfer learning.

More recent works are trying to find more efficient learning methods. Grill et al. [18] introduce a new training loss without using negative samples by utilizing non-parametric updates of [16]. In comparison, Caron et al. [37] remove the non-parametric updates and redefines positive and negative terms in the contrastive loss as computed features based on cluster assignment instead of using features from a projection function [17] directly. These works [18], [37], [38] can reduce the large amount of batch size required in previous studies [17] that use members in a mini-batch as negatives.

The mentioned techniques work well for image retrieval and image representation learning tasks without using labels. Unlike these studies, we propose a novel self-supervised deep metric learning technique for embedding unlabeled pointsets, which is *not* domain-specific.

2.4 Data augmentation for representation learning

Data augmentation is a popular technique for increasing training data size. With the exploration of self-supervised and semi-supervised representation learning mentioned in the previous subsection, data augmentation has been used for learning representation by maximizing the similarity between augmentation views (i.e., contrastive learning). Data augmentation in computer vision tasks can be done by basic image processing operations (e.g., cropping, resizing, rotation, distortion, blurring). Dogus et al. [39] proposed techniques for finding the optimal augmentation policies for each image dataset. Several works in representation learning studied in augmentation policies for image representation learning [17], [18], [37], [38], [40].

In NLP, data augmentation is a crucial factor in the success of state-of-the-art pre-trained language models. The augmentation technique called masked language modelling has been used in self-supervised training in BERT [41] and RoBERTa [23]. Xie et al. [42] proposed TF-IDF-based word replacement and used the back-translation method with a semi-supervised learning pipeline to achieve the state-of-the-art text classification performance.

Similar to graphs, You et al. [24] introduce graph augmentation techniques, e.g., node dropping for applying in a contrastive learning pipeline.

For other specific data domains, such as point clouds, Chen et al. [26] proposed PointMixup, a novel data augmentation method for point clouds using linear interpolation between two similar point clouds, based on their optimal assignments. This technique can increase the dataset size, which is applicable to any pointsets. However, linear interpolation might not generalize to all pointsets in other domains (e.g., set of words, set of images, set of graph nodes). We overcome this issue by proposing a general augmentation technique for pointsets based on swapping the elements of two similar pointsets. Furthermore, we apply the augmented pointsets for learning representations in our proposed self-supervised loss rather than only using augmentation to enlarge the training set.

2.5 Deep learning on sets

In our work, we make use of deep learning on pointsets. One of the first challenges of applying a neural network to process pointsets is the mismatch between the order invariance of pointsets and the order sensitivity of the network's input. An early attempt to address this challenge provides a specific convention to reorder the elements before putting them into an LSTM network [43]. Recent approaches introduce permutation invariance directly into the model. For example, DeepSets [44] uses an aggregation function to aggregate encoded features of each set member, or RepSet [2] represents pointsets using the output from a network flow matching algorithm. Another area of research aims to learn pointsets representation in an unsupervised fashion (i.e., set autoencoder [19], [20]).

One advancement in deep learning on pointsets is attention-based set encoders [45]. These networks apply a self-attention mechanism, i.e., Transformer [46], which helps capture the relationship between set elements. In our study, we apply the self-attention mechanism of Transformer and

DeepSets for encoding pointsets and learning a representation in a self-supervised way.

3 Problem Formulation

Consider three pointsets $x_a = \{x_{a1}, x_{a2}, x_{a3}, ...\}$, $x_b = \{x_{b1}, x_{b2}, x_{b3}, ...\}$, and $x_c = \{x_{c1}, x_{c2}, x_{c3}, ...\}$ represented as finite sets of vectors. Let $S(\cdot, \cdot)$ and $f(\cdot)$ be a distribution distance and an embedding function for pointsets, respectively. Our goal is to learn $f(\cdot)$ such that if $S(x_a, x_b) < S(x_a, x_c)$, then $||f(x_a) - f(x_b)|| < ||f(x_a) - f(x_c)||$. Specifically, the output representation $f(x) \in \mathbb{R}^F$ is located in an embedding space associated with the Euclidean metric. Given this embedding space, one can solve a retrieval task or classify a query input, e.g., with k-nearest neighbor algorithm. The main challenge is how to find $f(\cdot)$ that well generalizes to downstream tasks from *zero training labels*.

To achieve this, we try to learn our embedding function $f(\cdot)$ to resemble the behavior of a distribution distance function, which has been proven to be effective in pointset retrieval and classification. We consider a distance function known as Earth Mover's Distance (EMD) [1], [4]. The EMD is a distribution metric for any two pointsets, which calculates the minimum cost to transform one pointset to the other. To calculate the EMD, we first need to find the optimal assignment flow $F' = [f'_{ij}]$ to move set elements from x_a to x_b. The flow can be found using linear programming by minimizing a *work* based on an equation in Rubner et al. [4]:

$$\text{WORK}(x_a, x_b, F') = \sum_i \sum_j f'_{ij} d_{ij} \tag{1}$$

subject to the following conditions:

1) $f'_{ij} \geq 0$
2) $\sum_j f'_{ij} \leq w_{x_{ai}}$
3) $\sum_i f'_{ij} \leq w_{x_{bj}}$
4) $\sum_i \sum_j f'_{ij} = \min(\sum_i w_{x_{ai}}, \sum_j w_{x_{bj}})$,

where i and j state an entry of each element in pointsets x_a and x_b, respectively. w_x is the weight of each element in the pointset x, and d_{ij} is the ground distance, e.g., Euclidean distance between two elements in the two pointsets, x_{a_i} and x_{b_j}.

Then, we can calculate the EMD between two pointsets by using the solution from above optimal flow F' as:

$$\text{EMD}(x_a, x_b) = \frac{\sum_i \sum_j f'_{ij} d_{ij}}{\sum_i \sum_j f'_{ij}}, \tag{2}$$

With these formulations, EMD can be used to capture matching elements and the distance between two pointsets. For example, the study from M.J. Kusner et al. [1] shows that EMD can retrieve similar sets of word vectors accurately using k-nearest neighbors in the EMD space. They called a document retrieval framework that uses EMD as a distance metric as Word Mover's Distance (WMD). In the next section, we demonstrate how to utilize EMD to learn an embedding function in a self-supervised fashion and how to use EMD to augment pointsets.

4 Proposed Solution

In this section, we describe our proposed solution in detail. For ease of exposition, we use the problem of document classification as our narration framework. However, our solution can be applied to any learning problem in which data entries can be represented as pointsets.

Our key idea is to parameterize the embedding function as a permutation invariant neural network f_θ, or *Pointset2Vec* function, and use the triplet loss [7] with our *pseudo* labels to optimize for the embedding space. In the typical triplet loss optimization with supervision, each "anchor" document will be forced to be closer to a document belonging to the same class and farther away from a document of a different class, called the positive and negative, respectively. Our key contribution is a self-supervised method for identifying the positive and negative pairs associated with each anchor document without ground-truth labels. The overview of our proposed solution is shown in Fig. 1.

Inspired by the success of Earth mover's distance (EMD) [1] used for document classification, image, and multimedia retrieval tasks, we propose to use EMD for inferring the positive and negative pairs. In particular, a document with a low EMD to an anchor document will be labeled positive, and a document with a high EMD will be labeled negative. In addition, we introduce a re-weighting approach that emphasizes negative documents that are more likely to belong to different classes.

To enhance the performance of our loss, we also propose a new augmentation technique for pointsets, *PointSwap*. With PointSwap, we can extend triplet loss to minimize distances between anchor documents and augmented documents, similar to the concept of contrastive learning.

For the embedding function, we employ a neural network based on a state-of-the-art architecture in NLP, i.e., Transformer [46] and its self-attention mechanism which achieves permutation invariance suitable for representing pointsets [3], [44], [45].

4.1 Weighted Self-supervised EMD triplet loss

To train our network with triplet loss, we first need to form a set of document triplets, each of which consists of an anchor, a positive, and a negative as shown in Fig. 2. Given a mini-batch of size $\hat{n}$, we generate $\hat{n}$ training triplets by taking each document x_i in the mini-batch as the anchor and taking the closest document to each anchor in the EMD space as the positive x_i^p. For the negative, we follow the semi-hard negative mining strategy from [7]. This strategy mines a negative based on document embedding from $f(\cdot)$ by choosing a document closest to the anchor in the embedding space but also farther away from the anchor than the positive in the embedding space. Formally, the negative x_i^n is a document closest to the anchor in the embedding space that satisfies: $\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 < \left\|f(x_i^a) - f(x_i^n)\right\|_2^2$. With these training triplets, we then train our mapping function, parametrized by a neural network f with the following triplet loss:

$$L(X) = \frac{1}{\hat{n}} \sum_{i=0}^{\hat{n}} \left[\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 - \left\|f(x_i^a) - f(x_i^n)\right\|_2^2 + \alpha \right]_+ , \tag{3}$$

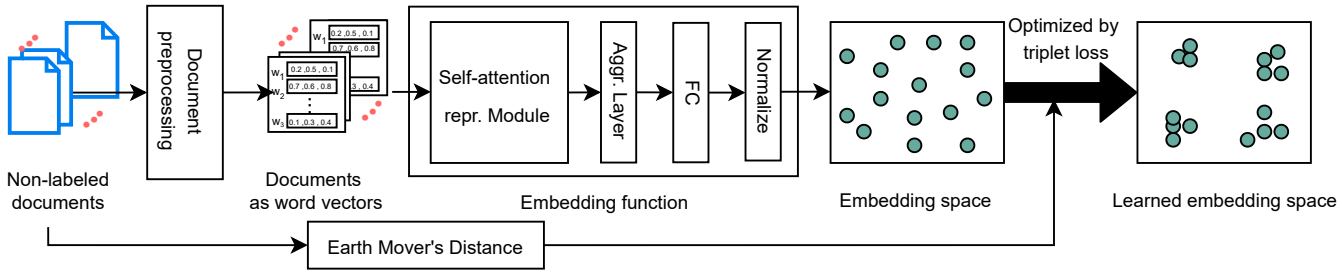


Fig. 1. The overall architecture of our proposed method using document input for illustrative purposes. We first process each unlabeled document into a set of word vectors, then embed this set into our embedding space with a permutation invariant neural network. This embedding network is optimized using our triplet loss in a self-supervised fashion through our automatic triplet selection.

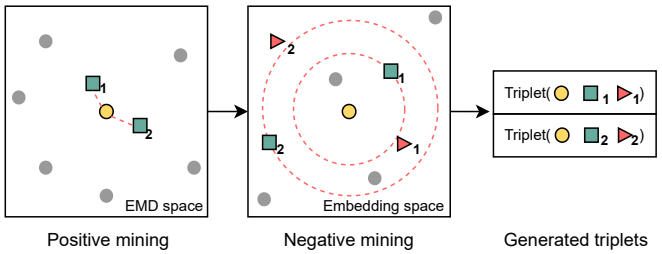


Fig. 2. An illustration of triplet selection. Given an anchor point (yellow circle), the closest point to this anchor in EMD space is chosen as the positive (green rectangle). Then, the negative is the closest point to the anchor that is at least as far way as the distance between the anchor and the positive (dotted ring) in the embedding space.

where X is the set of documents in each mini-batch, $\hat{n}$ is the total number of triplets in each mini-batch, and α is a hyper-parameter denoting the desired margin between the anchor-positive and anchor-negative distances in the embedding space.

One of the challenges of semi-hard negative sampling without supervision is that we cannot be certain that those negative documents that are quite close to the positives truly belong to different classes. This could be especially problematic in our self-supervised setting. One way to alleviate the effect of mislabeling is to prioritize negative documents that are more likely to come from different classes, which can be captured by higher EMDs to their anchors [1]. To achieve this, we simply re-weight the negative pair distances so that the more likely negatives are moved farther away than the less likely ones. In particular, our modified triplet loss has the following form:

$$L(X) = \frac{1}{\hat{n}} \sum_{i=0}^{\hat{n}} \left[\left\| f(x_i^a) - f(x_i^p) \right\|_2^2 - w_i \left\| f(x_i^a) - f(x_i^n) \right\|_2^2 + \alpha \right]_+, \tag{4}$$

where w_i is an exponential decay re-weighting factor for the i^{th} triplet and is defined as

$$w_i = \exp\left(-\frac{EMD(x_i^a, x_i^n)}{2(c\sigma)^2} \right), \tag{5}$$

where σ is the standard deviation of all pairwise EMD distances in each mini-batch and c is a scaling hyperparameter for σ. Negative documents whose EMDs are higher will be given lower w_i, and in order to satisfy the margin constrain of the triplet loss, the network needs to make these negatives even farther away from their anchors.

We call this loss function *Weighted self-supervised triplet loss* or *WSSET*. Surprisingly, such a simple modification also consistently outperforms the standard version given in Eq. (3). In addition, our loss function can also utilize with distance functions other than EMD. Please refer to our experimental section for results.

4.2 PointSwap

Following state-of-the-art representation learning studies [17], [18], [38], learning the function $f(\cdot)$ by maximizing the similarity between augmentation views of each training data could improve the representation performance of a model. However, performing data augmentation on generic pointsets remains largely unexplored, unlike other structured data, such as 2D images. There exist a study of data augmentation on 3D pointsets (i.e., point clouds) [26], but this study has not been verified in other pointset domains such as documents and graphs, where their elements are high dimensional features.

In our framework, we introduce a new augmentation technique for *generic* pointset by simply *modifying some elements in the set*. Given a document x, we generate an augmented document x' by modifying some random word vectors $x_i \in x$ such that the modified document x' still looks similar to x (i.e., it is still of the same class). To achieve this, we first find *another document* relevant to x to perform the modification. This document will be selected based on the closest document to x in the EMD space, i.e., x^p. Finding such a x^p requires computing the EMD, which also yields an optimal assignment flow F' between each word vector from x to x^p. We perform the modification to some x_i by finding its closely related words ${x^p}_j$ based on the assignment flow. Given $F' = [f'_{ij}]$, the modification to each x_i, performed only with a probability ω, is done by setting x_i to some word vector ${x^p}_j$ whose assignment flow $x_i \rightarrow x_j^p$ is non-zero. Formally, we obtain x' based on modifying each x_i by:

$$\begin{aligned} x' &= \text{PointSwap}(x, x^p) \\ &= \{{x'}_0, {x'}_1, {x'}_2, ..., {x'}_k\}, \text{ where} \\ {x'}_i &= \begin{cases} {x^p}_j & \text{if } f'_{ij} > 0 \text{ and } u_i < \omega \\ x_i & \text{otherwise,} \end{cases} \end{aligned} \tag{6}$$

where $u = u_0, u_1, ..., u_k$ is a set of uniform samples from $[0, 1]$ and ω is a probability threshold for performing word swapping. We call this randomly swapping operation *PointSwap*. By performing PointSwap on x with the most

relevant document x^p, we can ensure that the augmented result x' will also be similar to the original one. In addition, PointSwap is also universal and could be used with any data entry that can be represented as a finite pointset.

4.2.1 Combining PointSwap to Self-supervised triplet loss

With our proposed PointSwap and the augmented pointsets x', we could improve the self-supervised representation learning performance of our proposed triplet loss (Eq. (4)). In particular, we can force the model to learn a representation by maximizing the similarity between original and augmented documents, which leads to more generalized and robust representations, as shown in contrastive learning studies ([16], [17], [18], [37]). Specifically, we add the augmented version of the anchor document $x_i'^a$ into Eq. (4) and assign it as pseudo-positive for forcing the model to bring the anchor and augmented anchor close together:

$$L(X) = \frac{1}{2\hat{n}} \sum_{i=0}^{\hat{n}} \left[\left\| f(x_i^a) - f(x_i^p) \right\|_2^2 - w_i \left\| f(x_i^a) - f(x_i^n) \right\|_2^2 + \alpha \right]_+ + \left[\left\| f(x_i^a) - f(x_i'^a) \right\|_2^2 - w_i' \left\| f(x_i^a) - f(x_i'^n) \right\|_2^2 + \alpha \right]_+, \tag{7}$$

For the negative term $x_i'^n$, we also follow the semi-hard negative mining by choosing a document closest to the anchor in the embedding space that satisfies:$\left\| f(x_i^a) - f(x_i'^a) \right\|_2^2 < \left\| f(x_i^a) - f(x_i'^n) \right\|_2^2$. The anchor-augmented negative pair distances is also re-weighted by w_i' in Eq. (5). With PointSwap and this loss extension, the learned model can improve representation performance and outperform the state of the art in document classification in our experiments.

4.3 Self-attention permutation invariant neural network

To model the embedding function $f(x)$ in Eq. (4) and (7) as a permutation invariant neural network, we follow a general approach [44] that decomposes the final embedding of a set into a sum of individual embeddings of its elements. By the commutative property of addition or aggregation, the final embedding is thus permutation invariant.

In our model, the neural network that is applied to each individual element, a word vector w in our case, is modeled using a self-attention module [46], which we will refer to as *Encoder*. The structure of our model is also illustrated in Fig. 1. Our final embedding function is

$$f(x) = \rho \left(\sum_{w \in x} \textsc{Encoder}(w) \right), \tag{8}$$

where ρ is modeled using three fully-connected layers. Our self-attention module, *Encoder*, is a stack of Transformer's encoders [46] similar to the one proposed in a state-of-the-art model BERT [41] in NLP. These encoders help the network understand the inner-set correlation of the documents, and by stacking them multiple times, we allow the network to model more complex relationships between each word in the documents.

5 Application: Transferring the Learned Representations

With our proposed method in the previous section, we can construct a neural network that can map any *unlabeled* pointsets into an embedding space. However, real-world problems usually have various kinds of downstream tasks and datasets. Some of the available datasets could be provided with labels, whether sufficient for training in a supervised way or not. We then analyze the capability of our learned representation that could be *fine-tuned* on the given labels to power up the performance of those downstream tasks.

Instead of training a model for solving a targetted task from scratch, which could result in overfitting, fine-tuning a pre-trained representation on the given labels could alleviate the issue, as shown in studies on data domains, e.g., computer vision [17], [18] or NLP [23], [41]. Similar to these studies, we hypothesize that our learned pointsets representation is transferable to downstream tasks. Thus, we introduce our method in terms of a pre-trained model for using in training strategies as shown in Fig. 3. We split a strategy to construct a pre-trained model based on the dataset used in the pre-training step into two ways:

- *Internal-data pre-training:* To solve the target downstream task with a given target dataset, we construct a pre-trained model on the same dataset in a self-supervised fashion before fine-tuning on its labels.
- *External-data pre-training:* We construct a self-supervised pre-trained model based on our loss for two steps: pre-training a model on an extra dataset first, then pre-training the model again on the target dataset before fine-tuning with the target labels.

With these strategies, we could construct a pre-trained model for any generic pointsets that could enhance the performance of any downstream tasks on pointsets. We also formulate a set of hypotheses forming the basis for our experimental studies in the later subsection.

1) We expect our learned representation based on our proposed loss to be informative and can be used to evaluate downstream tasks on pointsets, with comparable performance to EMD [1], [4].
2) We expect that the representation from both internal and external-data pre-trained models should benefit the fine-tuning step, especially on datasets with insufficient labeled data or in semi-supervised settings. Their performance should be better than a model trained from scratch.
3) We expect the external-data pre-trained model to outperform other competitors by benefiting from the learned representation on an external dataset with a similar distribution to the target dataset.

We have also provided an external-data pre-trained model on product review texts, which is ready for fine-tuning with documents in the similar domain. The pre-trained model and its implementation details will be clarified in the next section.

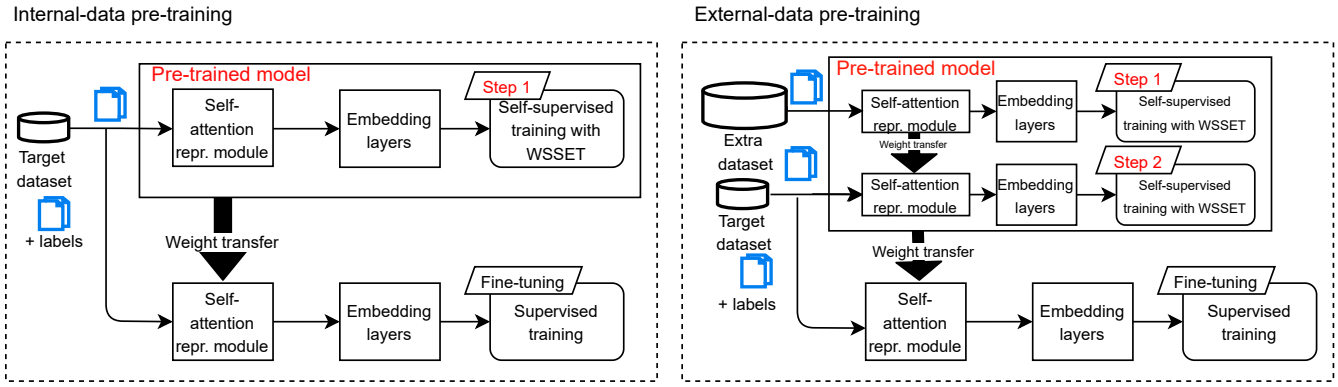


Fig. 3. Illustrations of two training strategies. The left figure shows internal-data pre-training pipeline which constructs a pre-trained model from the target dataset without using labels, before fine-tuning the model with labels. The right figure shows external-data pre-training which first constructs a pre-trained model from an additional dataset without using labels, then followed by self-supervised pre-training with the target dataset again, which is similar to the steps of internal-data pre-training.

6 Experimental Setup

6.1 Datasets

To evaluate our proposed method and extension ideas proposed in the previous section, we use two types of pointsets data, i.e., documents and graphs. We use four datasets for document retrieval and classification tasks [1], [2]: *BBCSports*, *Twitter*, *Recipe*, and *Amazon* with the original train/test splits and randomly choose 10% from the train set as the validation set. Collectively, these datasets cover a wide range of characteristics such as *easy and small*, *short and imbalanced*, *and challenging*. We use classification accuracy as the main evaluation metric similar to the practice in the literature [1], [2] and Recall@K metric to measure the retrieval performance.

- *BBCSports* is a sports article dataset containing 517 training and 220 testing articles classified into 5 types of sports. The average number of samples per class is 147.4; the maximum number of words is 452. This dataset represents an *easy and low-resource scenario*.
- *Twitter* [47] is a set of tweets collected from a social media platform, Twitter, which contains 2,176 training and 932 testing tweets. Each tweet is labeled with 3 types of sentiments, i.e., positive, negative, or neutral. It has an average of 1036 samples per class and 26 maximum number of words. This dataset represents a *short, challenging, and imbalanced dataset*. (The neutral class alone takes up 60%).
- *Recipe* is a dataset of cooking instructions which contains 3,059 training and 1,311 testing instructions with the maximum number of words of 360. Each instruction is classified into one of 15 regions of origin with an average of 291.33 samples per class. This dataset represents *a challenging dataset* since most of the method cannot achieve a classification accuracy more than 60%.
- *Amazon* is a dataset of Amazon product reviews containing 5,600 training and 2,400 testing reviews, classified into 4 types of products. The average samples per class is 2,000 with the maximum number of words of 884. This dataset represents *an easy and long document dataset*.

TABLE 1
Graph datasets statistics

Datasets	Properties			
	#Graphs	*#Classes*	*Nodes(Avg.)*	*Types*
MUTAG	188	2	17.93	Molecules
IMDB-B	1500	2	19.77	Social
NCI1	4110	2	29.87	Molecules
PROTEINS	1113	2	39.06	Bioinformatics

For graph datasets, we use four graph kernel datasets from TUDatasets [21]: *MUTAG*, *IMDB-B*, *NCI1*, and *PROTEINS*. The statistical information of each dataset is shown in Table 1. We split the dataset into 10 folds to perform cross-validation. For the main evaluation metric, we report mean classification accuracy and standard deviation of 10 splits as same as other studies [2], [24].

For point cloud dataset, we use ModelNet40 [22] dataset for classification tasks with the original train/test splits.

6.2 Implementation

6.2.1 Data preparation

The input document is converted into a set of word vectors using GoogleNews Word2Vec model with a feature size of 300. We also append the word frequency to the feature vector, resulting in a feature size of 301. Additionally, we zero-pad the dimension corresponding to the number of words in each document so that each document has the same number of words, i.e., the maximum number of words in all documents.

For graphs, we represented each graph in the datasets as an adjacency matrix (a set of nodes). In addition, we zero-pad the dimension of each node to have the same number of edges (i.e., the maximum number of edges in all graphs).

For point clouds, we follow PointMixup's [26] setup by sampling 1,024 points, normalizing them to a unit sphere, and randomly jittering each point.

6.2.2 Model architecture

Our *Encoder* in Section 4.3 uses a stack of 5 Transformer's encoders, each of which uses 7 attention heads in the multi-head attention module for feature of size 301 and a feed-forward layer with 1,000 hidden nodes. The ρ function is modeled with three fully-connected layers of sizes 512, 256,

and 64, where only the first two have ReLU activations. We also normalize the last layer using L2 normalization.

We trained our network with a batch size of 64 using Adam optimizer with a learning rate of 10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$ for 1,000 epochs in each experiment and use the model that has the best accuracy on the validation set for testing. For Eq. 4 and Eq. 7 losses, we use $\alpha = 0.1$, $c = 7$, and $\omega = 0.5$. For fine-tuning, we use semi-hard negative mining supervised triplet loss [7] with $\alpha = 0.1$.

To benchmark our models, we use a distance-weighted k-nearest neighbor classifier (kNN) with $k = 10$ to predict labels in experiments of classification task in Section 7.

6.3 Variations of our proposed models

In this section, we explain variations of our models and their use cases in experimental studies. We will explain each variation and its settings based on two different use cases; unsupervised tasks and supervised tasks.[1]

6.3.1 Models for unsupervised tasks

In this use case, we compare the following proposed model with competitors in an unsupervised representation learning task:

- *Ours*: This model is trained using weighted self-supervised EMD triplet loss (Eq. (4)) without using provided labels.

6.3.2 Models for supervised tasks

For comparing with state-of-the-art supervised pointset classification methods, we proposed the following variations:

- *Ours+CE/TL*: We use this model as a baseline by training the model using supervised triplet loss (TL) [7] or cross-entropy loss (CE) with the provided labels.
- *Ours+Int.*: We construct this model by fine-tuning training labels with the Internal-data pre-trained model mentioned in Section 5. As shown in Fig. 3, we first perform self-supervised pre-training (Eq. (4)) with the target dataset before fine-tuning with its labels. In fine-tuning step, we re-initialize the weights of the fully connected layers ρ after Encoder in Eq. (8). Then, we fine-tune the model using supervised triplet loss with the provided labels. This model can achieve a better performance than training with the supervised losses based on the provided labels directly.
- *Ourst+Ext.*: We construct this model by fine-tuning training labels with the External-data pre-trained model mentioned in Section 5. We use the same setting as Ours+Int. except that we first pre-train on an external dataset in a self-supervised fashion and re-initialize the weights of the fully connected layers before pre-training with a target dataset. In our experimental section, we used *Consumer Reviews of Amazon Products* dataset on Kaggle as the external pre-train data. This dataset contains 28,332 reviews of the Amazon products. We compare this model to supervised competitors with external data pre-training, e.g., language models.

For the ease of understanding, we summarize each variation and its required steps in Table 2.

In addition, we can choose whether to train each model pointset augmentation, or apply our proposed PointSwap (PS), or use competitors': PointMixup (PM) [26] or PointDrop (PD). PointMixup is specifically designed to augment point clouds using linear interpolation. While PointDrop, inspired by node dropping methods in graph augmentation [24], is a method that randomly drops each element in each set with a certain probability. In the case of training a model by applying the augmented pointset, we use Eq. (7) as a loss function with x'^a from the chosen augmentation method in all self-supervised training stages.

TABLE 2
Variation of our proposed models

Model name	External pre-training	Internal pre-training	Supervised fine-tuning
Ours		✓	
Ours+CE/TL			✓
Ours+Int.		✓	✓
Ours+Ext.	✓	✓	✓

7 Experimental results

7.1 Comparison with unsupervised and self-supervised approaches

In this section, we evaluate our proposed self-supervised learning technique by using nearest neighbor classification task against a traditional distances for computing nearest neighbors (i.e., WMD [1] and Chamfer Distance (CD) [27]) and these following unsupervised/self-supervised representation learning approaches:

- We used the unsupervised representation learning technique proposed by Dosovitskiy et al. [14], called *Exemplar*. We applied DeepSets to encode pointsets for the encoder instead of original implementation with CNN and image inputs.
- We implemented a direct EMD approximation loss technique that learns a space on which the Euclidean distance approximates EMD, called *Approx.EMD*, using the same encoder as ours with the following loss: $L(X) = \frac{1}{n^2}\sum_{i,j}^{n}(\text{EMD}(x_i, x_j) - \parallel f(x_i) - f(x_j) \parallel_2^2)^2$
- We also adopt the current state-of-the-art self-supervised representation learning, contrastive learning methods, called *Cont.* for a more extensive comparison. We implemented a competitor based on SimCLR [17] method using the same encoder as ours.

These competitors except Approx.EMD have been designed based on other applications with different losses and data preparation steps. For a fair comparison, we follow their original training steps but changed their feature encoder to be a baseline pointset encoder, e.g., DeepSets [44] and the self-attention encoder (see Section 4.3).

Table 3 shows that Ours with and without PointSwap outperforms CD, Exemplar, Approx.EMD, and variations of

1. Code and data available at https://github.com/vistec-AI/WSSET/

Cont. Ours and the variants perform better than WMD only on BBCSport. Nevertheless, WMD takes $O(n^3 \log n)$ for one distance calculation, while our method has a *905.9 times* lower computation cost. This will be further discussed in the retrieval performance experiments. Furthermore, Ours could beat WMD by fine-tuning strategies, which will be shown in the following subsection on supervised and semi-supervised experimental settings.

We found that using PointSwap as an augmentation method achieved higher accuracy than PointMixup, PointDrop, and no augmentation for both Ours and Cont. in most datasets. In particular, the differences between the performance of PointSwap and other augmentation settings in Cont. are significant statistically in all datasets. For Ours, PointSwap outperforms PointMixup for all datasets but has lower accuracy than PointDrop and no augmentation in Twitter dataset. However, Cont. with PointSwap still has poor performance compared to Ours and WMD. One possible reason is that the augmented pointsets from these methods may not be suitable for the contrastive learning pipeline. These issues are also shown in computer vision studies [17], [37], [40] as contrastive learning requires ablation experiments to find effective augmentation techniques.

Lastly, Approx.EMD has lower performance than our variations, as mentioned in Section 4 but still outperforms the other approaches.

TABLE 3
Comparison of our representation with unsupervised and self-supervised approaches

Methods	Aug	Datasets			
		BBCSport	*Recipe*	*Amazon*	*Twitter*
WMD [1]	-	**95.0**	**58.42**	**92.75**	**71.49**
CD [27]	-	80.00	42.48	83.41	66.20
Exemplar [14]	-	58.64	33.26	69.79	70.17
Approx.EMD	-	93.18	50.13	84.79	70.49
Cont.	-	69.09	39.21	74.67	68.26
Cont.	PD	78.64	41.95	78.04	70.02
Cont.	PM	78.00	40.35	76.75	69.21
Cont.	PS	80.45	44.7	83.71	70.17
Ours	-	96.36	51.87	92.17	**70.71**
Ours	PD	94.22	48.17	91.48	70.52
Ours	PM	96.36	52.10	92.25	68.88
Ours	PS	**97.27**	**52.40**	**92.37**	70.28

7.2 Comparison with supervised approaches

We evaluated the fine-tuning performance of internal and external-data pre-trained models by comparing our fine-tuning results in document classification tasks with state-of-the-art supervised learning methods for pointsets and text classification models.

For the comparison between methods that are trained without external data shown in the upper part of Table 4, we found that Ours+Int. outperforms all competitors, including DeepSets, Set transformer [45], RepSet [2], and fine-tuned contrastive learning model, i.e., Cont.+Int. In particular, using internal-data pre-trained models can boost the accuracy of training from scratch settings (i.e., Ours+TL) for all datasets. Interestingly, using Cont. pre-trained model, which has poor nearest neighbor accuracy in the unsupervised settings, gains better accuracy than Ours+TL on all datasets except for Recipe.

To compare Ours+Ext., we fine-tuned two state-of-the-art language models, $\text{BERT}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{BASE}}$, with raw text data provided in the dataset except for Recipe, which lacks raw text inputs. Ours+Ext. with PointSwap outperforms both competitors as reported in the lower part of Table 4 and also outperforms the fine-tuned contrastive learning model, Cont.+Ext.

These results demonstrate the effectiveness of our pre-trained model for both internal and external pre-training methods.

For the augmentation choices, pre-training with PointSwap achieves better accuracy than others for non-external and external settings, similar to the previous experimental results. However, using PointDrop and PointMixup shows negative transferring (i.e., has lower performance than non-transfer learning methods) on Recipe and Amazon datasets by getting lower accuracy than pre-training without augmentation.

We also found that Cont+Ext achieves lower performance than both Cont+Int and the model without transfer learning, Ours+TL. We hypothesize that the contrastive learning pipeline may not fit enough for pre-training a model on external, i.e., out-of-domain data.

TABLE 4
Comparison of our fine-tuned models' results with other supervised approaches

Methods	Aug	Datasets			
		BBCSport	*Recipe*	*Amazon*	*Twitter*
DeepSets+CE	-	97.7	58.88	94.95	75.42
DeepSets+TL	-	99.09	54.3	94.10	73.6
Ours+CE	-	97.27	61.00	94.30	76.6
Ours+TL	-	97.75	62.30	94.65	77.5
SetTrans [45]	-	95.82	57.46	92.82	72.21
RepSet [2]	-	98.00	61.43	94.71	74.58
Cont.+Int.	-	100.00	59.88	94.88	77.36
Cont.+Int.	PD	100.00	59.65	95.04	77.42
Cont.+Int.	PM	100.00	60.86	94.96	77.9
Cont.+Int.	PS	100.00	61.87	95.08	78.33
Ours+Int.	-	100.00	63.16	95.50	78.33
Ours+Int.	PD	100.00	59.42	95.33	77.47
Ours+Int.	PM	100.00	62.47	95.58	78.43
Ours+Int.	PS	**100.00**	**63.16**	**95.58**	**78.43**
$\text{BERT}_{\text{BASE}}$	-	99.54	-	95.21	75.85
$\text{RoBERTa}_{\text{BASE}}$	-	99.09	-	95.65	77.89
Cont.+Ext.	-	98.64	53.93	91.5	75.21
Cont.+Ext.	PD	99.09	51.85	94.36	76.17
Cont.+Ext.	PM	99.09	52.71	94.5	76.50
Cont.+Ext.	PS	99.09	56.83	94.54	76.72
Ours+Ext.	-	100.00	63.31	95.63	78.11
Ours+Ext.	PD	100.00	60.14	95.42	77.97
Ours+Ext.	PM	100.00	62.70	95.54	78.11
Ours+Ext.	PS	**100.00**	**63.31**	**95.71**	**78.33**

7.3 Experiments on semi-supervised learning settings

Inspired by previous experiments, we also evaluated the transferring performance of our pre-trained model by fine-tuning with partially-labeled datasets (i.e., semi-supervised setting). We simulated this setting by reducing a labeled training set into 1%, 3%, 5%, 7%, and 10% out of available training labels. We compare our methods with a distance-based baseline (i.e., WMD), RoBERTa, and state-of-the-art semi-supervised learning technique for text, UDA [42].

Fig. 4 shows trends of classification accuracy and the number of labels used in the fine-tuning step. Using our

pre-trained model's representation can beat competitors in most settings. In particular, Ours+Int. with PointSwap outperforms no augmentation version and can achieve 4.94% difference from RoBERTa in Twitter dataset with 1% labels and 5% difference in BBCSport dataset at 5% labels. However, RoBERTa got higher accuracy than ours with 1% labels in Amazon dataset. This could be because Amazon reviews are similar to the pre-trained dataset used in the language model.

Interestingly, both internal and external pre-trained models have comparable results in all settings. We hypothesize that using self-supervised representation from the dataset in the same domain, regardless of external data, is sufficient for solving supervised tasks with small numbers of labeled data.

One remarkable finding is that fine-tuning such a small number of labels with our pre-trained model could make Ours+Int. and Ext. variants perform better than WMD. This could be an example to demonstrate our representation's strength, which can be fine-tuned with any target problem even with a low resource of labeled data. On the other hand, fine-tuning WMD on a target problem is non-trivial.

7.4 Experiments on graph datasets

We also evaluate our self-supervised learning method for pointsets on graphs classification tasks. We show a comparison against the supervised pointset encoder, RepSet, and state-of-the-art self-supervised graph representation learning, GraphCL [24].

With our self-supervised representation, we achieve comparable results to both RepSet and GraphCL on all four datasets, as shown in Table 5. It is also interesting to note that compared to the method without augmentation, PointSwap gains mean accuracies at 2.61%, 1.35%, 10.93%, and 6.38% in MUTAG, IMDB-B, NCI1, and PROTEINS datasets, respectively. At the same time, PointMixup has lower accuracy than no augmentation in MUTAG and IMDB-B, while PointDrop achieves a comparable result to PointSwap. This finding is similar to the study [24] that already demonstrates the benefits of node dropping technique on graph learning problems.

Our pre-trained model also benefits fine-tuning with partially-labeled graph datasets as shown in Table 6. We sample 10% of the training labels in the graph datasets for fine-tuning the internal-data pre-trained model. The results show that our pre-trained model can boost the supervised baseline (i.e., Ours+CE) accuracies. Pre-training our model using PointSwap still outperforms PointMixup, and has comparable results with PointDrop. Moreover, the results of Ours+Int. with PointSwap also has comparable results with GraphCL for all datasets.

These results demonstrate the ability of our learning method, both proposed loss and PointSwap, which could be applied to not just a set of word vectors but also other domains of pointsets.

7.5 Experiments on point cloud datasets and Chamfer Distance

Not only graph datasets, but we also additionally evaluated our proposed loss in the point cloud classification task. We also show a study on Chamfer Distance (CD), which has remarkable performance in measuring similarity between two point clouds in several studies.

TABLE 5
Comparison of mean classification accuracy and standard deviation conducted on a 10-fold cross-validation split graph datasets with other set-based and graph-based approaches

Method	Datasets			
	MUTAG	*IMDB-B*	*NCI1*	*PROTEINS*
RepSet	88.63±0.86	**71.46±0.91**	-	73.04±0.86
GraphCL	86.80±1.34	71.14±0.44	**77.87±0.41**	**74.39±0.45**
Ours	91.66±3.46	70.45±2.38	65.39±0.76	68.50±4.87
Ours+PD	91.68±3.04	71.9±3.48	75.58±1.38	74.24±3.67
Ours+PM	90.53±5.24	70.35±3.08	70.53±2.63	73.57±2.53
Ours+PS	**93.05±4.52**	71.75±2.89	76.32±1.53	74.88±2.13

TABLE 6
Comparison of mean classification accuracy and standard deviation conducted on a 10-fold cross-validation split with other general-based and graph-based approaches on graph datasets with 10% label propagation

Methods	Datasets (10% labeled)	
	NCI1	*PROTEINS*
Ours+CE	58.89±3.15	69.70±3.25
GraphCL [24]	**74.63±0.25**	**74.17±0.34**
Ours+Int.	71.02±2.43	70.41±2.04
Ours+Int.+PD	73.52±2.50	73.10±3.16
Ours+Int.+PM	71.50±2.69	72.03±2.32
Ours+Int.+PS	73.68±2.54	72.16±2.15

Table 7 shows that using CD with kNN achieves better accuracy than EMD. By comparing the distance calculation time per point cloud pair, CD also has a faster computation complexity of $O(n^2)$ while EMD is $O(n^3 \log n)$.

We also conducted an experiment to show the effectiveness of varying the distance function (i.e., CD and EMD) used for mining triplets in our loss (Fig. 2). In this experiment, we follow PointMixup [26] by changing the encoder to PointNet++ [48] and train with our loss proposed in Section 4.1 and fine-tuning the pre-trained model with given labeled data using triplet loss. We found that learning representation with EMD results in higher accuracy than CD. Similar to a semi-supervised setting with 10% labels, fine-tuning a pre-trained model on EMD also leads to higher accuracy.

Interestingly, using CD in our loss could not achieve better accuracy as computing nearest neighbors by the kNN algorithm. Nevertheless, CD still has a lower calculation complexity than EMD, plus the accuracy gap is small. Thus, using CD in the pipeline can help reduce the training time with some performance trade-offs. More information on the trade-off between CD and EMD can be found in Appendix A.

By comparing each augmentation setting, PointSwap could boost classification accuracy from no augmentation in self-supervised and semi-supervised settings. PointSwap has slightly lower accuracy than PointMixup [26] in both settings. Nevertheless, PointMixup is a specialized method for point cloud data, which does not generalize well to pointsets in other domains as shown in the previous experiments. While PointSwap could achieve classification accuracies gaining on all selected three domains of pointsets (i.e., documents, graphs, and point clouds). Another interesting point is that PointSwap could also boost the performance

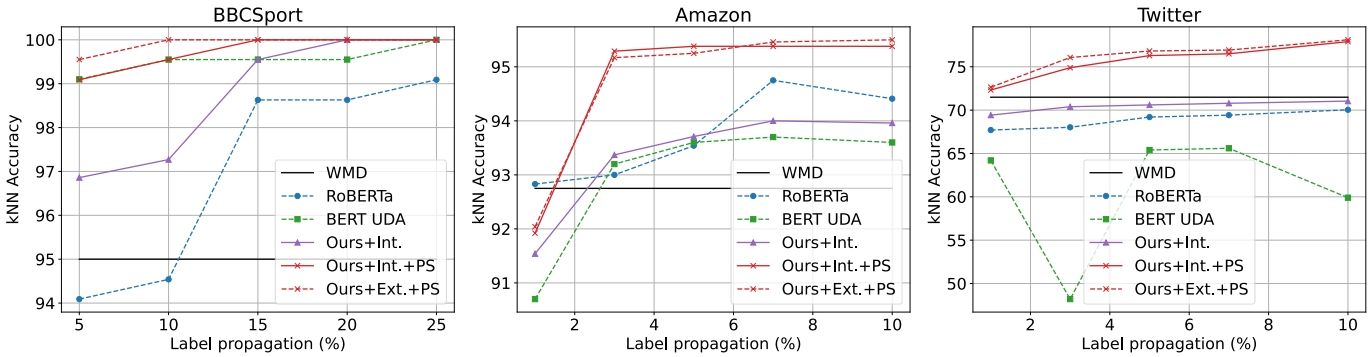


Fig. 4. This figure reports kNN accuracies on semi-supervised settings explained in Section 7.3, which simulately reduce the label propagation of available training labels used in the fine-tuning step. Solid lines represent approaches that do not use any external data, while dashed lines represent approaches that do.

TABLE 7
Effect of varying augmentation and distance used for mining triplet in our proposed loss. The experiments were conducted on the ModelNet40 dataset in unsupervised and semi-supervised settings by fine-tuning a pre-trained model with 10% label propagation.

Methods	Aug	Distance	Unsup	Semi-sup (10%)
Distance	-	EMD	**77.87**	-
		CD	75.85	-
TL	-	-	-	59.60
Ours	-	EMD	60.12	67.24
		CD	58.99	66.20
	PM	EMD	**63.08**	**68.11**
		CD	62.39	67.26
	PS	EMD	61.06	67.54
		CD	60.45	66.89

of a generic supervised point cloud classification baseline without augmentation and outperforms PointMixup [26]. This experiment can be found in Appendix B.

7.6 Experiments on approximation and unsupervised tasks

In this subsection, we demonstrate how our pre-trained model can be extended to improve other target tasks, i.e., kernel density estimation in EMD space. To achieve this, we first need to fine-tune our self-supervised pre-trained model to output a representation in Euclidean space, where each sample has a distance to another sample as equal to EMD$(\cdot)$. We could do this by fine-tuning with direct EMD approximation loss, which has been used to train the baseline, *Approx.EMD* presented in Section 7.1 previously.

Table 8 shows the comparison between Approx.EMD and our fine-tuned models, Ours+Int.(EMD) and Ours+Ext.(EMD). Mean squared error (MSE) shows the error between our approximated distance and the ground truth EMD values. Both of our models can achieve lower MSE than the baseline, except Ours+Ext.(EMD) that got higher error on Twitter and Ours+Int.(EMD) on Recipe. These results demonstrate the benefits of transferring learned representation to other objectives, which is not limited to only the supervised fine-tuning as shown in the previous experiments.

To evaluate the performance of distance approximated from our model, we use density estimation as the downstream task. We compare the discrepancy (relative error) of the approximation from models against the Gaussian kernel density estimation from the ground truth EMD, using the following formula: $G(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$ with σ of 1. The estimations are calculated from kNNs with the k value of 10.

Similar to MSE, both of our models got lower discrepancy than the baseline, except Ours+Ext.(EMD) on Twitter. This clearly shows that the benefit of having a more accurate EMD approximation method can translate to more accurate kernel density estimation results.

TABLE 8
Comparison of mean squared error between our approximated distance and the ground truth EMD values and density estimation discrepancy with neural-network based EMD approximation approaches

Methods	Datasets			
	BBCSport	*Recipe*	*Amazon*	*Twitter*
Mean squared error				
Approx.EMD	.0100	.0037	.0347	.0035
Ours+Int.+PS(EMD)	.0100	.0054	.0112	**.0030**
Ours+Ext.+PS(EMD)	**.0093**	**.0029**	**.0086**	.0041
Mean density estimation discrepancy (%)				
Approx.EMD	6.249	2.206	20.072	2.846
Ours+Int.+PS(EMD)	6.195	2.679	4.581	**2.534**
Ours+Ext.+PS(EMD)	**6.023**	**2.078**	**4.091**	3.247

7.7 Experiments on retrieval performance

Following pointsets retrieval [13] and deep metric learning [15], [34] works, we evaluated the retrieval performance of learned representation by measuring Recall@K on top-K similarity search and showing computation time compared to distance-based and neural network-based methods.

Table 9 shows Recall@K between competitors mentioned in Section 7.1 and our methods. Ours and Ours with PointSwap outperform Approx.EMD in all settings and also achieve higher recall over WMD at lower K on BBCSports and Amazon datasets. Interestingly, PointDrop shows comparable retrieval results to PointSwap and no augmentation on Twitter dataset. This finding is similar to classification accuracies evaluated in Section 7.1. For Recipe, our methods also achieve comparable Recall@K to WMD. This demonstrates the ability to learn a good embedding space with zero labels using our self-supervised loss.

In some settings, Ours + PointSwap might have lower retrieval performance (i.e., Recall@k) than Ours. This is because PointSwap, which is based on modifying some elements in the original set using *another relevant set*. In particular, a selected relevant set from the EMD space might have a chance to belong to a different class from

the original one, resulting in an augmented set with some biases from other classes. Hence, our model learned from these biased datasets might generate biased representation (i.e., have similar attributes to representations of different classes). Giving a query to perform searching in such a biased representation pool might result in more incorrect retrieval results appearing in top k.

With these challenges regarding the robustness of our PointSwap, a more robust set augmentation technique could be an interesting topic for further investigation.

We also show the average query time on documents and graphs database. For documents, we choose Amazon dataset to evaluate our approach against WMD [1], which is a special case of the classic EMD ($O(n^3 \log n)$) [4], threshold EMD ($O(n^2 \log n)$) [28] and three neural network-based methods, BERT [41], RoBERTa [23], and RepSet [2]. For graphs, we do a comparison with graph neural network-based method, GraphCL [24] on NCI1 dataset. A time complexity comparison between each competitor has been provided in Appendix A. All of the studies were conducted on Intel Xeon E5-2698 v4 2.20GHz with 20-core multiprocessing for distance calculation and NVIDIA Tesla V100 for neural network inference.

Fig. 5 shows that our approach is the fastest method among the competitors. However, BERT and RoBERTa use a tokenizer to process raw text, which is faster than Word2Vec in other approaches. Nonetheless, both BERT and RoBERTa take a longer time to encode a query to a feature vector than our model since our model has 4.9M parameters while BERT and RoBERTa have 110M and 125M parameters, respectively. At the same time, RepSet, which uses a set-based neural network, also has a longer network inference time than ours.

In the distance calculation phase, the neural network approaches (i.e., ours and language models) which use Euclidean distance are *39.625 times* faster than Threshold EMD and *905.9 times* than WMD.

While Fig. 6 also shows that our approach has a better inference time than GraphCL, which yields better graph classification performance but is *29.5 times* slower than ours.

These results demonstrate the potential of our self-supervised learning method, which could be used for pointset retrieval with comparable performance and faster computing time than EMD-based approaches and neural network-based competitors.

7.8 Study on pointsets data augmentation

This section shows a comprehensive study of PointSwap by evaluating it against other pointset augmentation techniques.

7.8.1 Effect of augmentation hyperparameters

We first study the behavior of PointSwap, PointDrop, and PointMixup, regarding their controllable parameter ω in Eq. 6 for PointSwap, dropout probability for PointDrop, and mix ratio λ for PointMixup. We varied ω and λ from 0.1, 0.3, 0.5, 0.7, to 1.0, where lower values of ω and λ mean that the augmented set x'^a is mostly similar to its original x^a, and higher ω and λ mean that each element in x'^a has been mostly modified by elements in x^p based on the optimal assignments. To evaluate this, we conducted experiments on the document (BBCSport and Recipe) and graph (MUTAG and IMDB-B) datasets using unsupervised pointset classification tasks.

TABLE 9
Comparison of Recall@K with other unsupervised and self-supervised approaches

BBCSports	**R@5**	**R@15**	**R@30**	**R@45**	**R@60**
WMD [1]	7.28	24.31	**46.98**	**66.79**	**84.05**
Approx.EMD	8.01	21.89	36.17	46.25	54.19
Ours	**9.10**	**25.64**	45.59	58.97	78.11
Ours+PD	7.87	22.46	39.85	51.81	61.18
Ours+PM	8.81	24.59	42.61	55.03	64.17
Ours+PS	8.95	25.51	45.42	57.78	76.27
Recipe	**R@25**	**R@50**	**R@100**	**R@200**	**R@400**
WMD [1]	**4.45**	**8.30**	**15.20**	**27.18**	**48.11**
Approx.EMD	3.54	6.42	11.54	20.75	37.13
Ours	4.19	7.73	13.82	23.96	40.65
Ours+PD	3.06	5.76	10.65	19.60	36.12
Ours+PM	4.02	7.36	13.16	23.08	39.62
Ours+PS	4.20	7.73	13.87	23.99	40.38
Amazon	**R@50**	**R@100**	**R@300**	**R@500**	**R@700**
WMD [1]	6.58	12.87	36.09	**57.19**	**76.63**
Approx.EMD	5.13	9.77	26.60	39.76	49.98
Ours	**6.92**	**13.40**	35.95	52.24	63.48
Ours+PD	6.56	12.80	35.65	52.15	62.48
Ours+PM	6.79	13.10	34.82	50.95	60.26
Ours+PS	6.89	13.29	**36.24**	53.17	64.20
Twitter	**R@50**	**R@100**	**R@200**	**R@300**	**R@500**
WMD [1]	**5.90**	**11.62**	**22.89**	**33.99**	**55.85**
Approx.EMD	5.73	11.19	21.93	32.37	53.35
Ours	5.78	11.35	22.00	32.58	53.56
Ours+PD	5.79	11.32	22.19	32.61	53.64
Ours+PM	5.77	11.37	22.16	32.58	53.26
Ours+PS	5.79	11.37	22.19	32.64	53.60

Table 10 shows that $\omega = 0.5$ is the best parameter. In addition, using $\omega = 0.1$ gains the lowest accuracy, which could imply that using low ω would have lower accuracy than the no augmentation version. For PointDrop, using dropout probabilities of 0.5 and 0.7 on document and graph datasets can achieve the best accuracies, respectively. While in PointMixup, λ has peak accuracy between $\lambda = 0.5$ and 0.7. Based on these results, we use $\omega = 0.5$, dropout probability = 0.5 for documents and 0.7 for graphs, and $\lambda = 0.5$ to compare with other competitors for the following experiments.

7.8.2 Comparison between PointSwap and baseline augmentation methods

We then compare PointSwap, PointDrop, PointMixup, and other baseline pointsets augmentation methods against a model without augmentation.

We implemented three variants of PointSwap based on two techniques used in the method described in Section 4.2: i) Nearest EMD: We select one similar pointset of the anchor pointset based on its nearest one in EMD space, i.e., positive. ii) Optimal assignment: This technique swaps each element in the pair based on its optimal assignments from EMD calculation (see Eq. (1)). The three variants are as follow:

- The first variant selects a pair for anchor randomly and swaps each element without assignment.
- The second variant also selects a pair randomly but swaps each element based on an assignment.

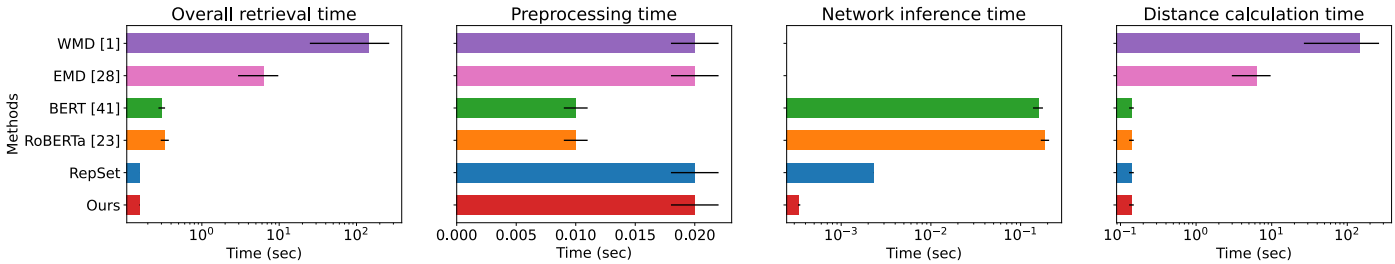


Fig. 5. These graphs show the average time consumption for each step in similarity search on Amazon dataset from a given query. Note that we use a log scale on the overall retrieval time and distance calculation time. Ours is the fastest approach in the overall retrieval time and even faster than the state-of-the-art neural network *BERT* while achieving with comparable accuracy.

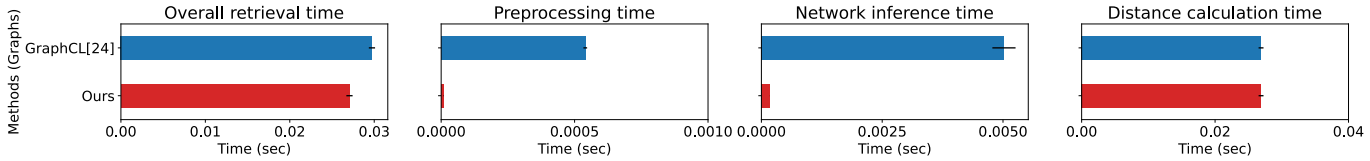


Fig. 6. These graphs show the average time consumption for each step in similarity search on NCI1 dataset from a given query. Ours is faster than the state-of-the-art graph neural network *GraphCL*.

- The third variant selects a pair based on its nearest EMD of the anchor but swaps each element without assignment.

Each variant was trained using Eq. (7) for unsupervised settings and semi-supervised settings using 5% of labeled data from document datasets. There is also a summary in Table 11 for showing how each variant differs from the other.

Table 11 also shows a summary of accuracy gained relative to the no augmentation version (i.e., Eq. (4)) for each augmentation technique. We found that PointSwap achieves the highest accuracy among other methods in all datasets in the unsupervised setting. We also observed that the first variant obtains the lowest accuracy, while the second, third variants, and PointDrop perform differently on different datasets. Interestingly, these variants and PointDrop, which obtain lower performance in the unsupervised setting, can achieve positive gains on some datasets in the semi-supervised settings. These performance gains demonstrate the benefit of pre-trained models with augmentation, regardless of their poor performance when directly evaluating the embedding space in the self-supervised learning step.

Nevertheless, PointSwap and PointMixup, which did not negatively affect the accuracy evaluated in the self-supervised learning step, can achieve better accuracy for all datasets in fine-tuning steps than the no augmentation baseline.

In Table 12, we show samples of augmented words using each augmentation technique on document datasets. The sampled augmented words from the three baselines show that all of them are unrelated. PointMixup can generate the same words with the interpolated word vectors but also unrelated words. Interestingly, we found that PointSwap's augmented words are mostly synonyms or somehow related words. We hypothesize that using synonyms instead of interpolated vectors could lead to better representation. This hypothesis could also happen in graphs, as replacing the anchor node with its similar node (i.e., PointSwap) has better results than *shifting* edges (i.e., PointMixup) as shown in the graph experiments in Section 7.4.

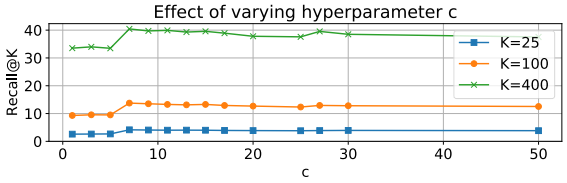


Fig. 7. Figure shows the effectiveness of our re-weighting negative terms on recall@K by tuning the hyperparameter c in Eq.5.

7.9 Hyperparameter study

We conducted a study to demonstrate the effectiveness of re-weighting negatives by varying the hyperparameter c in Eq.5 on Recipe, which is the most challenging dataset among all 4 datasets. Higher c means less effect of re-weighting. We observe that for all Recall@K, $c = 7$ to $c = 13$ have significantly better recall rates than higher c values as shown in Fig. 7. The results show that prioritizing negative pairs based on EMD could help the network learn with samples that are more likely to belong to different classes.

8 Conclusion

We have proposed a self-supervised deep metric learning approach for pointsets, which enables effective representation learning on unlabeled datasets. The key idea of our method is the use of EMD to generate pseudo labels for triplet loss training, and a re-weighting technique that encourages larger separation between certain negative pairs deemed more likely to come from different classes. We have also proposed a new augmentation technique for pointsets, called PointSwap, which randomly swaps an element in one pointset and another element in a similar set. Our experiments show that our learned representation can be used for fine-tuning with downstream tasks and produce results superior to other supervised competitors, including an NLP-specific approach based on a state-of-the-art language model. In addition, our method is shown to be the

TABLE 10
Effect of varying augmentation methods' probability threshold ω and λ on document and graph datasets. Mean classification accuracy and standard deviation conducted on a 10-fold cross-validation split are shown for graph datasets.

ω/λ	0.1	0.3	0.5	0.7	1.0
PointSwap (PS)					
BBCsport	89.18	94.27	**97.27**	93.66	**96.36**
Recipe	47.02	50.99	**52.40**	51.14	**51.87**
MUTAG	89.99±4.68	90.01±3.58	**93.05±4.52**	88.61±4.53	**91.66±3.46**
IMDB-B	69.33±2.68	**71.25±2.34**	**71.75±2.89**	70.11±2.03	70.45±2.38
PointDrop (PD)					
BBCsport	86.36	91.65	**94.22**	93.97	-
Recipe	42.33	44.52	**48.17**	46.08	-
MUTAG	88.89±4.24	90.00±4.11	85.55±5.37	**91.68±3.04**	-
IMDB-B	69.33±2.68	69.20±1.56	69.7±0.87	**71.9±3.48**	-
PointMixup (PM) [26]					
BBCsport	87.26	93.82	**96.36**	93.00	**96.36**
Recipe	46.80	51.14	51.68	**52.10**	**51.87**
MUTAG	88.69±5.18	88.57±6.53	**90.53±5.24**	89.44±7.49	**91.54±2.98**
IMDB-B	68.55±3.75	69.50±3.61	**70.72±1.72**	70.35±3.08	**70.45±2.38**

TABLE 11
Comparison of accuracy gaining from no augmentation self-supervised learning between training with PointSwap and its other variations on unsupervised and semi-supervised settings.

Augmentation methods	**Nearest EMD**	**Optimal assignment**	**Datasets (Unsupervised)**				**Datasets (5% labeled)**			
			BBCSport	***Recipe***	***Amazon***	***Twitter***	***BBCSport***	***Recipe***	***Amazon***	***Twitter***
(1)	-	-	-12.27	-11.06	-13.25	-1.26	-3.82	-2.36	0.75	3.03
(2)	-	✓	-16.36	-5.72	-18.75	-0.65	-2.46	0.84	1	2.82
(3)	✓	-	-5.45	-7.17	-5.55	-0.94	-1.55	-0.84	0.58	2.82
PointDrop	-	-	-2.14	-3.70	-0.69	**-0.19**	**0.27**	0.69	1.07	2.82
PointMixup [26]	✓	✓	0.00	0.23	0.08	-1.83	**0.27**	2.52	**1.33**	1.53
PointSwap	✓	✓	**0.91**	**0.53**	**0.2**	-0.43	**0.27**	**2.75**	**1.33**	**3.25**

TABLE 12
The generated augmented word vectors using three baselines, PointMixup, and PointSwap

Datasets	**Original words**	**Augmentation methods**				
		(1)	***(2)***	***(3)***	***PointSwap***	***PointMixup [26]***
BBCSports	olympic	s	european	olympic	olympics	line
	congratulated	number	learn	personal	enjoyed	game
	bangladesh	liverpool	i	win	spinners	bangladesh
	refused	head	pieces	told	denies	refused
Recipe	remove	food	large	turns	leave	low
	bowl	cups	long	tea	tail	bowl
	above	noodles	-	bowl	low	sheet
	brown	increase	accomplishment	fix	gray	olives
Twitter	eyes	i	now	details	screen	screen
	antitrust	here	use	none	racketeering	microsoft
	own	with	-	by	personal	developed
	appstore	quit	slew	the	iphone	is

fastest among all the competitors. As future work, we plan to further generalize this approach to other pointsets data or other distance functions and investigate more robust set augmentation methods.

ACKNOWLEDGMENTS

This work was supported by PTT public company limited, SCB public company limited, and the Digital Economy Promotion Agency, Thailand, under depa Digital Infrastructure Fund for Private Investment (MOA 040/2562, MP-62-0003). The research of Cheng Long is supported by the Ministry of Education, Singapore, under its Academic Research Fund (Tier 2 Award MOE-T2EP20220-0011) and by the Nanyang Technological University Start-Up Grant from the College of Engineering under Grant M4082302.

REFERENCES

[1] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *ICML*, 2015.

[2] K. Skianis, G. Nikolentzos, S. Limnios, and M. Vazirgiannis, "Rep the set: Neural networks for learning set representations," in *AISTATS*, vol. 108, pp. 1410–1420, PMLR, 2020.

[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CVPR*, pp. 77–85, 2017.

[4] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *IJCV*, vol. 40, pp. 99–121, 2000.

[5] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," *CVPR*, pp. 4004–4012, 2016.

[6] F. Xu, W. Zhang, Y. Cheng, and W. Chu, "Metric learning with equidistant and equidistributed triplet-based loss for product image search," in *WWW*, WWW '20, p. 57–65, 2020.

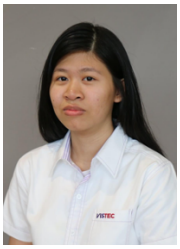
[7] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CVPR*, Jun 2015.

[8] J. Deng, J. Guo, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *CVPR*, pp. 4685–4694, 2018.

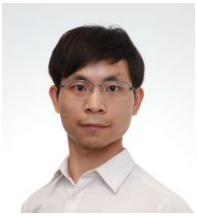
[9] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, "Adacos: Adaptively scaling cosine logits for effectively learning deep face representations," *CVPR*, pp. 10815–10824, 2019.

[10] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *ArXiv*, vol. abs/1703.07737, 2017.

[11] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *NIPS*, 2016.
[12] D. Yao, G. Cong, C. Zhang, and J. Bi, "Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach," in *ICDE*, pp. 1358–1369, 2019.
[13] P. Arsomngern, C. Long, S. Suwajanakorn, and S. Nutanong, "Self-supervised deep metric learning for pointsets," in *ICDE*, pp. 2171–2176, 2021.
[14] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. A. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *TPAMI*, vol. 38, 2016.
[15] M. Ye, X. Zhang, P. C. Yuen, and S.-F. Chang, "Unsupervised embedding learning via invariant and spreading instance feature," *CVPR*, 2019.
[16] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," *CVPR*, Jun 2020.
[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, pp. 1597–1607, 2020.
[18] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - a new approach to self-supervised learning," in *NeurIPS*, vol. 33, 2020.
[19] M. Probst, "The set autoencoder: Unsupervised representation learning for sets," 2018.
[20] Y. Zhang, J. S. Hare, and A. Prügel-Bennett, "Deep set prediction networks," in *NeurIPS*, 2019.
[21] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," in *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.
[22] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, pp. 1912–1920, 2015.
[23] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
[24] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *NeurIPS*, vol. 33, 2020.
[25] J. Yang, P. Ren, D. Zhang, D. Chen, F. Wen, H. Li, and G. Hua, "Neural aggregation network for video face recognition," in *CVPR*, pp. 5216–5225, 2017.
[26] Y. Chen, V. T. Hu, E. Gavves, T. Mensink, P. Mettes, P. Yang, and C. G. M. Snoek, "Pointmixup: Augmentation for point clouds," *Lecture Notes in Computer Science*, 2020.
[27] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *IJCAI*, pp. 659–663, 1977.
[28] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *ICCV*, pp. 460–467, 2009.
[29] M.-H. Jang, S.-W. Kim, C. Faloutsos, and S. Park, "A linear-time approximation of the earth mover's distance," in *CIKM*, 2011.
[30] I. Assent, A. Wenning, and T. Seidl, "Approximation techniques for indexing the earth mover's distance in multimedia databases," in *ICDE*, pp. 11–11, IEEE, 2006.
[31] Y. Tang, L. H. U, Y. Cai, N. Mamoulis, and R. Cheng, "Earth mover's distance based similarity search at scale," *VLDB*, 2013.
[32] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, vol. 1, pp. 539–546 vol. 1, 2005.
[33] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *SIMBAD*, 2015.
[34] B. Yu and D. Tao, "Deep metric learning with tuplet margin loss," *ICCV*, pp. 6489–6498, 2019.
[35] Y. Gao, Y.-F. Li, S. Chandra, L. Khan, and B. Thuraisingham, "Towards self-adaptive metric learning on the fly," WWW '19, p. 503–513, 2019.
[36] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," *CVPR*, pp. 3733–3742, 2018.
[37] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2020.
[38] X. Chen and K. He, "Exploring simple siamese representation learning," in *CVPR*, June 2021.
[39] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *CVPR*, June 2019.
[40] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," in *NeurIPS*, 2020.
[41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019.
[42] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," 2019.
[43] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," *CoRR*, vol. abs/1511.06391, 2016.
[44] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, "Deep sets," in *NIPS*, 2017.
[45] 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 *ICML*, 2018.
[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017.
[47] Sanders and N.J., "Sanders-twitter sentiment corpus," *Sanders Analytics LLC.*, 2011.
[48] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017.



Pattaramanee Arsomngern Pattaramanee Arsomngern is currently a Ph.D. student at the School of Information Science and Technology, Vidyasirimedhi Institute of Science and Technology (VISTEC), Thailand. She received the B.Eng. degree from King Mongkut's University of Technology Thonburi (KMUTT), Thailand, in 2019. Her research interest is deep representation learning.



Cheng Long Cheng Long (S'11-M'15) is currently an Assistant Professor at the School of Computer Science and Engineering, Nanyang Technological University. From 2016 to 2018, he worked as a lecturer at Queen's University Belfast, UK. He received his PhD degree from the Hong Kong University of Science and Technology, Hong Kong, in 2015, and his BEng degree from South China University of Technology, China, in 2010. His research interests are broadly in data management, data mining and big data analytics.

Supasorn Suwajanakorn Supasorn Suwajanakorn is currently a lecturer at the School of Information Science and Technology (VISTEC), Thailand. He received his PhD from the University of Washington and B. Eng. from Cornell University. His research interests lie in the intersection of computer vision, deep learning, and computer graphics.

Sarana Nutanong Sarana Nutanong received the Ph.D. degree from the University of Melbourne, Australia. He held an assistant professor position at the City University of Hong Kong. He is currently an Associate Professor with the School of Information Science and Technology, Vidyasirimedhi Institute of Science and Technology (VISTEC), Thailand. His research interests include scientific data mining, information retrieval, and natural language processing.

Supplementary Materials: Towards Pointsets Representation Learning via Self-Supervised Learning and Set Augmentation

Pattaramanee Arsomngern, Cheng Long, Supasorn Suwajanakorn, and Sarana Nutanong

We provide additional clarifications and experiments in this supplementary material. We present a study of an alternative distance choice, Chamfer Distance, in our loss function in Appendix A. In Appendix B, we demonstrate the effectiveness of our proposed augmentation technique, PointSwap, in a standard supervised point cloud classification task. Finally, Appendix C details the time complexity of each competitor shown in the runtime comparison (Section 7.7).

APPENDIX A
STUDY OF CHAMFER DISTANCE IN DIFFERENT DOMAINS OF DATASETS

Chamfer distance (CD) [1] is a well-known technique for measuring the similarity between two non-weighted distributions, especially for point clouds and computer vision-related problems. CD only requires $O(n^2)$ to compute the directed distance from set a to set b, which is faster than our selected distribution distance (i.e., EMD with $O(n^3 \log n)$). With these advantages of CD, we conducted an experiment to show the effectiveness of CD on various pointset datasets.

Table 1 shows a comparison between CD and EMD for doing k-nearest neighbor classification with $k = 10$ on documents, point clouds, and graphs datasets. We find that CD has better accuracies than EMD on point clouds, as expected, and one graph dataset. However, EMD outperforms CD on all four document datasets. This might be because EMD can utilize the weights of the set members, i.e., word frequencies, in the calculation step, while CD can not.

By combining with the experiment results in Section 7.5 in the main paper, we find that both CD and EMD have comparable performance when applying our proposed self-supervised loss. Thus, using CD in the pipeline can help reduce the training time with some performance trade-offs. On the other hand, EMD can be preferred if each set member has an associated weight.

TABLE 1
kNN accuracies for each competitor, EMD and CD

Datasets	Distances	
	EMD	CD
BBCSports	95.00	80.00
Recipe	58.42	42.48
Amazon	92.75	83.41
Twitter	71.49	66.20
ModelNet40	75.85	77.87
MUTAG	63.82	65.95
IMDB-B	63.00	50.10

APPENDIX B
APPLYING POINTSWAP IN SUPERVISED POINT CLOUD CLASSIFICATION PIPELINE

We show the results of using PointSwap on the ModelNet40 dataset in a supervised point cloud classification pipeline in the PointMixup study [2]. In this pipeline, we follow their open-access implementation on GitHub[1] by using PointNet++ [3] encoder without their proposed ManifoldMixup. We use $\lambda = 0.4$ for PointMixup and $\omega = 0.2$ for PointSwap. Note that running their provided implementation has lower accuracies than the results reported in their paper [2].

Table 3 shows that both PointMixup and PointSwap outperform the no augmentation variant. Interestingly, PointSwap can achieve better accuracy than PointMixup, which is consistent with comparisons in documents and graphs datasets shown in Sections 7.1, 7.2, and 7.4. This suggests that PointSwap can be used with any finite pointset.

APPENDIX C
TIME COMPLEXITIES OF EACH COMPETITOR

Table 2 shows the time complexities of all competitors that have been evaluated on the retrieval experiments in Section 7.7. We split the comparison into three stages of retrieval; network inference, distance calculation, and database lookup. We also provide the total complexity of all three steps. For network inference, BERT [6], RoBERTa [7], and Ours use the same Vaswani et al.'s Transformer [10], which has $O(n^2)$ time complexity, where n is the number of members in sets. RepSet [8] uses MLP and bipartite

- *P. Arsomngern, S. Suwajanakorn, and S. Nutanong are with School of Information Science and Technology, Vidyasirimedhi Institute of Science and Technology, Thailand. E-mail: {pattaramanee.a_s19, supasorn.s, snutanon}@vistec.ac.th*
- *C. Long is with School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail:c.long@ntu.edu.sg*

1. https://github.com/yunlu-chen/PointMixup

TABLE 2
Time complexities of each approach in each retrieval process.

Methods	Time complexity per query			
	Network inference	*Distance calculation*	*Database retrieval*	*Total*
WMD [4]	-	$O(n^3 \log(n))$	$O(D)$	$O(Dn^3 \log(n))$
EMD [5]	-	$O(n^2 \log(n))$	$O(D)$	$O(Dn^2 \log(n))$
BERT [6]	$O(n^2)$	$O(n)$	$O(D)$	$O(Dn^2)$
RoBERTa [7]	$O(n^2)$	$O(n)$	$O(D)$	$O(Dn^2)$
RepSet [8]	$O(nm)$	$O(n)$	$O(D)$	$O(Dnm)$
GraphCL [9]	$O(n^2)$	$O(n)$	$O(D)$	$O(Dn^2)$
Ours	$O(n^2)$	$O(n)$	$O(D)$	$O(Dn^2)$

TABLE 3
Comparison of classification accuracies with other point cloud augmentation choices

Methods	Accuracy
No augmentation	91.25
PointMixup [2]	91.34
PointSwap	91.61

matching algorithm, which takes $O(nm)$, where m is the number of edges between set members and their proposed hidden sets. In the worst scenario, where hidden sets are the same size as the query set, m would be equal to n^2. In the case of GraphCL [9], it uses GIN [11] architecture, which has graph convolutional operations with time complexity $O(m)$ [12] (i.e., n^2). We can conclude that all competitors, including ours, have the same inference time complexity, except RepSet.

All mentioned neural network competitors could generate a feature vector in the Euclidean space, which takes only $O(n)$ to calculate the distance between two features.

WMD [4] and threshold EMD [5], which do not require network inferencing, have the distance calculation complexities $O(n^3 \log(n))$ and $O(n^2 \log(n))$, respectively. By performing a set retrieval on a database with D entries, WMD and approximate EMD have the largest computation complexity among the others.

REFERENCES

[1] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *IJCAI*, pp. 659–663, 1977.

[2] Y. Chen, V. T. Hu, E. Gavves, T. Mensink, P. Mettes, P. Yang, and C. G. M. Snoek, "Pointmixup: Augmentation for point clouds," *Lecture Notes in Computer Science*, p. 330–345, 2020.

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017.

[4] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *ICML*, 2015.

[5] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *ICCV*, pp. 460–467, 2009.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019.

[7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[8] K. Skianis, G. Nikolentzos, S. Limnios, and M. Vazirgiannis, "Rep the set: Neural networks for learning set representations," in *AISTATS*, vol. 108, pp. 1410–1420, PMLR, 2020.

[9] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *NeurIPS*, vol. 33, pp. 5812–5823, Curran Associates, Inc., 2020.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017.

[11] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *ICLR*, 2019.

[12] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.