Self-supervised Multi-view Learning via Auto-encoding 3D Transformations

XIANG GAO and WEI HU, Wangxuan Institute of Computer Technology, Peking University, China
GUO-JUN QI, OPPO US Research Center, United States

3D object representation learning is a fundamental challenge in computer vision to infer about the 3D world. Recent advances in deep learning have shown their efficiency in 3D object recognition, among which view-based methods have performed best so far. However, feature learning of multiple views in existing methods is mostly performed in a supervised fashion, which often requires a large amount of data labels with high costs. In contrast, self-supervised learning aims to learn multi-view feature representations without involving labeled data. To this end, we propose a novel self-supervised framework to learn Multi-View Transformation Equivariant Representations (MV-TER), exploring the equivariant transformations of a 3D object and its projected multiple views that we derive. Specifically, we perform a 3D transformation on a 3D object and obtain multiple views before and after the transformation via projection. Then, we train a representation encoding module to capture the intrinsic 3D object representation by decoding 3D transformation parameters from the fused feature representations of multiple views before and after the transformation. Experimental results demonstrate that the proposed MV-TER significantly outperforms the state-of-the-art view-based approaches in 3D object classification and retrieval tasks and show the generalization to real-world datasets. The code is available at https://github.com/gyshgx868/mvter.

CCS Concepts: • Computing methodologies → Shape representations; Object recognition;

Additional Key Words and Phrases: Self-supervised learning, multi-view learning, transformation equivariant representation

ACM Reference format:
Xiang Gao, Wei Hu, and Guo-Jun Qi. 2023. Self-supervised Multi-view Learning via Auto-encoding 3D Transformations. ACM Trans. Multimedia Comput. Commun. Appl. 20, 1, Article 19 (September 2023), 23 pages. https://doi.org/10.1145/3597613

1 INTRODUCTION

3D object representation has become increasingly prominent for a wide range of applications, such as 3D object recognition and retrieval [11, 33, 45, 47–49, 57, 69, 72, 74]. Recent advances in Convolutional Neural Network (CNN)-based methods have shown their success in 3D object recognition and retrieval [11, 57, 72, 74]. One important family of methods are view-based methods,
which project a 3D object into multiple views and learn compact 3D representation by fusing the feature maps of these views for downstream tasks. Feature learning of multiple views in existing approaches is mostly trained in a supervised fashion, hinging on a large amount of data labels that prevents wide applicability. Hence, self-supervised learning is in demand to alleviate the dependencies on labels by exploring unlabeled data for the training of multi-view feature representations in an unsupervised or (semi-)supervised fashion.

Many attempts have been made to explore self-supervisory signals at various levels of visual structures for representation learning, which leverages the input data itself as supervision and benefits almost all types of downstream tasks [43]. Recently, many approaches have sought to learn transformation equivariant representations to further improve the quality of self-supervised representation learning. Transformation Equivariant Representation learning assumes that representations equivarying to transformations are able to encode the intrinsic structures of data such that the transformations can be reconstructed from the representations before and after transformations [51]. Following Hinton’s seminal work on learning transformation capsules [26], a variety of approaches have been proposed to learn transformation equivariant representations [9, 10, 13, 16, 32, 38, 50, 53, 55, 56, 64, 75]. Specifically, Zhang et al. [75] propose to learn unsupervised representations of single images via Auto-Encoding Transformations (AET) by decoding transformation parameters from the learned representations of both the original and transformed images. Further, Gao et al. [13] extend transformation equivariant representations to graph data that are irregularly structured (e.g., 3D point clouds) and formalize graph transformation equivariant representation learning by auto-encoding node-wise transformations in an unsupervised manner. Nevertheless, these works focus on transformation equivariant representation learning of a single modality, such as 2D images or 3D point clouds.

In this article, we propose to learn Multi-View Transformation Equivariant Representations (MV-TER) by decoding the 3D transformations from multiple 2D views, aiming to capture the intrinsic structures of 3D objects. This is based on our proven equivariant transformations of a 3D object and its projected multiple 2D views—multi-view transformation equivariance. That is, when we perform 3D transformations on a 3D object, the 2D views projected from the 3D object via fixed viewpoints will transform equivariantly, i.e., the feature representation of each 2D view projected from the transformed 3D object is transformed from the feature representation of the corresponding 2D view projected from the original 3D object via the same viewpoint by a homomorphism of the 3D transformation, as proved in Section 3.1.2. In contrast to previous works where 2D/3D transformations are decoded from the original single image/point cloud and transformed counterparts, we exploit the equivariant transformations of a 3D object and the projected 2D views linked by the projection operation. We propose to decode 3D transformations from multiple views of a 3D object before and after transformation, which is taken as self-supervisory regularization to enforce the learning of intrinsic 3D representation. By estimating 3D transformations from the fused feature representations of multiple original views and those of the equivariantly transformed counterparts from the same viewpoints, we enable the accurate learning of 3D object representation even with a limited amount of labels.

Specifically, we first perform a 3D transformation on a 3D object (e.g., point clouds, meshes) and render the original and transformed 3D objects into multiple 2D views with a fixed camera setup. Then, we feed these views into a representation learning module to infer representations of the multiple views before and after transformation, respectively. A decoder is set up to predict the applied 3D transformation from the fused representations of multiple views before and after transformation. We formulate multi-view transformation equivariant representation learning as a regularizer along with the loss of a specific task (e.g., classification) to train the entire network end-to-end. Experimental results demonstrate that the proposed method significantly
outperforms the state-of-the-art view-based models in 3D object classification and retrieval tasks.

Our main contributions are summarized as follows:

- We propose **Multi-View Transformation Equivariant Representations (MV-TER)** to learn 3D object representations from multiple 2D views that transform equivariantly with the 3D transformation in a self-supervised fashion.
- We derive the **multi-view transformation equivariance**, i.e., when we perform 3D transformations on a 3D object, the 2D views projected from the 3D object via fixed viewpoints will transform equivariantly.
- We formalize the MV-TER as a self-supervisory regularizer to learn the 3D object representations by decoding 3D transformation from fused features of projected multiple views before and after the 3D transformation of the object.
- Experiments demonstrate the proposed method outperforms the state-of-the-art view-based methods in 3D object classification and retrieval tasks under various 3D transformations in a self-supervised fashion.

## 2 RELATED WORKS

In this section, we review previous works on self-supervised representation learning, transformation equivariant representations, as well as multi-view-based neural networks.

### 2.1 Self-supervised Representation Learning

Many self-supervised learning approaches have been proposed to train deep neural networks using self-supervised signals, which can be derived from data themselves without being manually labeled. The self-supervised model can be categorized into three categories [43]: generative-based, contrastive-based, and adversarial-based. Generation-based methods aim to train an encoder to encode the input data into an explicit vector and a decoder to reconstruct the input data from the vector [31, 61]. Contrastive-based methods often maximize the agreements between the augmented views of the same image in an embedding feature space, while avoiding the mode collapse of the embedded features by maximizing the disagreements between negative examples constructed from different images [3, 6, 7, 23, 27, 28, 62]. Adversarial-based methods train an encoder-decoder network to generate fake samples and a discriminator to distinguish them from real samples [17].

### 2.2 Transformation Equivariant Representations

Many approaches have been proposed to learn equivariant representations, including transforming auto-encoders [26], equivariant Boltzmann machines [32, 56], equivariant descriptors [53], and equivariant filtering [55]. Lenc et al. [38] prove that the AlexNet [34] trained on ImageNet learns representations that are equivariant to flip, scaling, and rotation transformations. Gens et al. [16] proposed an approximately equivariant convolutional architecture, which utilizes sparse and high-dimensional feature maps to deal with groups of transformations. Dieleman et al. [10] show that rotation symmetry can be exploited in convolutional networks for effectively learning an equivariant representation. Dieleman et al. [9] extend this work to evaluate other computer vision tasks that have cyclic symmetry. Cohen et al. [8] proposed group equivariant convolutions that have been developed to equivary to more types of transformations. The idea of group equivariance has also been introduced to the capsule nets [39] by ensuring the equivariance of output pose vectors to a group of transformations. To generalize to generic transformations, Zhang et al. [75] proposed to learn unsupervised feature representations via **Auto-Encoding Transformations (AET)** by estimating transformations from the learned feature representations.
of both the original and transformed images. Qi et al. [50] extend the AET by introducing a variational transformation decoder, where the AET model is trained from an information-theoretic perspective by maximizing the lower bound of mutual information. Wang et al. [64] extend the AET to Generative Adversarial Networks (GANs) for unsupervised image synthesis and representation learning. Gao et al. [13] proposed the GraphTER model that extends the AET to the graph domain and formalized graph transformation equivariant representation learning by auto-encoding node-wise transformations. In contrast to the AET and GraphTER, we focus on representations equivarying to projected 3D transformations onto multiple 2D views by estimating the applied 3D transformation, which exploits our derived multi-view transformation equivariance.

2.3 3D Object Learning

We review related works on 3D object learning, including volumetric-based methods, point-based methods, and view-based methods.

2.3.1 Volumetric-based Methods. These methods usually voxelize a point cloud into 3D grids and then apply 3D Convolutional Neural Networks on the volumetric representation for 3D representation learning [18]. Wu et al. [69] proposed to represent a geometric 3D shape as a probability distribution of binary variables on a 3D voxel grid using a Convolutional Deep Belief Network and constructed a large-scale 3D CAD model dataset ModelNet that attracts a large number of scholars to invest in 3D representation learning. Maturana et al. [45] proposed an architecture called VoxNet by integrating a volumetric Occupancy Grid representation with a supervised 3D Convolutional Neural Network for 3D object learning. However, these methods are unable to scale well to dense data due to the computational complexity that grows cubically with the resolution. To tackle this issue, hierarchical and compact structures [36, 52, 65] for the volumetric data have been proposed to reduce the computational cost. Riegler et al. [52] proposed to hierarchically partition the input point cloud using a set of unbalanced octrees, which reduces the memory allocation and enables deeper networks without compromising resolution. Wang et al. [65] proposed an octree-based convolutional neural network for 3D shape analysis that takes advantage of the sparsity of the octree representation to enable compact storage and fast computation. Le et al. [36] proposed a 3D convolutional network that integrates both point and grid representations to better represent the local geometry details.

2.3.2 Point-based Methods. Point-based methods belong to another family of 3D point cloud learning, which directly feed the unstructured and unordered 3D points to deep neural networks. PointNet [47] is a pioneer method that directly takes 3D point clouds as the input to learn the features of each point independently, while PointNet++ [49] introduces a hierarchical architecture to capture local geometric structures from the neighborhood of each point. Local structures of point clouds have also been exploited by methods such as PointCNN [41], PCNN [2], Relation-Shape CNN [44], and PointConv [68]. PointCNN [41] generalizes CNN by leveraging spatially local correlation from data represented in point clouds. PCNN [2] defines convolution of functions over point clouds, aiming to translate the volumetric convolution to arbitrary point clouds using extension and restriction operators. Relation-Shape CNN (RS-CNN) [44] extends regular grid CNN to irregular configuration for point cloud analysis. PointConv [68] trains multi-layer perceptrons on local point coordinates to approximate continuous weight and density functions in convolutional filters, which makes point sets permutation-invariant and translation-invariant. In addition, Graph Convolutional Neural Networks (GCNNs) have been applied to point clouds by treating points as nodes and constructing appropriate graphs such as nearest-neighbor graphs to learn feature representations [60, 63, 66, 76].
2.3.3 View-based 3D Object Learning. View-based methods project 3D objects (e.g., point clouds, meshes) into multiple views and extract view-wise features receptively via CNNs and then fuse these features as the descriptor of 3D objects. Su et al. [57] first proposed a multi-view convolutional neural network (MVCNN) to learn a compact descriptor of an object from multiple views, which fuses view-wise features via a max pooling layer. Qi et al. [48] introduce a new multi-resolution component into MVCNNs and improve the classification performance. Several subsequent works proposed to fuse multiple view-wise features into an informative descriptor for 3D objects. Feng et al. [11] proposed a group-view convolutional neural network (GVCNN) framework to fuse view-wise features using a grouping strategy. Yu et al. [74] proposed a multi-view harmonized bilinear network (MHBN) to learn 3D object representation by aggregating local convolutional features through the proposed bilinear pooling. To take advantage of the spatial relationship among views, Han et al. [20] and Han et al. [19] proposed to aggregate the global features of sequential views via attention-based RNN and CNN, respectively. Kanezaki et al. [30] proposed to learn global features by treating pose labels as latent variables. Yang et al. [72] proposed a relation network to connect corresponding regions from different viewpoints. Jiang et al. [29] propose a Multi-Loop-View Convolutional Neural Network (MLVCNN) for 3D object retrieval by introducing a loop normalization to generate loop-level features. Wei et al. [67] design a view-based graph convolutional network (GCN) framework to aggregate multi-view features by investigating relations of views. Yu et al. [73] tackle the 3D object representation learning from the perspective of set-to-set matching, where set-to-set matching kernels are introduced to highlight pairs consisting of relevant patches. Gao et al. [14] systematically evaluate the performance of deep learning features in view-based 3D model retrieval, which provides an important reference for other researchers. Gao et al. [15] propose a novel multi-level view associative convolution network for view-based retrieval, where the relationship exploration of multi-view images, the fusion of different images, and the feature discrimination learning are realized in a unified end-to-end framework.

Our method belongs to the family of view-based methods, as we also project 3D objects to 2D views for feature learning. Different from the above methods, we formalize the MV-TER as a self-supervisory regularizer optimized along with other losses to facilitate the learning of feature representations for various downstream tasks by ensuring transformation equivariance, which enforces the learning of intrinsic 3D representations from multiple 2D views. The proposed method is general in that the MV-TER regularizer can be easily plugged into view-based models to learn powerful representations.

3 METHOD

In this section, we first provide a comprehensive presentation of the proposed MV-TER method in Section 3.1 and then a detailed elaboration of the network structure in Section 3.2.

3.1 The Formulation

We first define transformations of 3D objects and projections onto multiple views in Section 3.1.1. Then, we prove the multi-view transformation equivariance in Section 3.1.2 and formulate the model in Section 3.1.3. Based on the formulation, we introduce two transformation decoding schemes in Section 3.1.4. Further, some analysis and discussion of the proposed MV-TER are provided in Section 3.1.5.

3.1.1 3D Transformations and Projections onto Views. 2D views are projections of a 3D object from various viewpoints. Formally, given a 3D object \( X \in \mathbb{R}^{n \times 4} \) consisting of \( n \) points\(^1\) and a 3D

\(^1\)In projective geometry, any point \( x_i = [x, y, z] \) in \( X \) is augmented to \([x, y, z, 1]\) in the homogeneous coordinate system.
transformation distribution $\mathcal{T}$, we sample a transformation $t \sim \mathcal{T}$ and apply it to $X$:

$$\hat{X} = t(X).$$

(1)

In this article, we consider 3D transformations that can be represented by matrices, such as linear transformations, affine transformations (e.g., rotation, shearing), or other types of changes applied to the 3D object for data augmentation. Please note that 3D translation only affects the positions of objects in 2D views, while 3D scaling affects the sizes of the 2D views. Neither of these transformations introduces changes in the object shape in 2D views. Therefore, in our subsequent applications, we only consider three types of 3D transformations: rotation, shearing, and combinations of these two transformations. The transformed 3D object can be rewritten as

$$\hat{X} = t(X) = (AX^T)^T,$$

(2)

where $A \in \mathbb{R}^{4 \times 4}$ denotes a 3D transformation matrix.

We project $X$ onto 2D views from $m$ viewpoints, denoted as $\mathcal{V} = \{V_1, \ldots, V_m\}$, i.e.,

$$V_i = p_i(X) = K_i [R_{c,i} \mid t_{c,i}] X^T, \quad i = 1, \ldots, m,$$

(3)

where $p_i : \mathbb{R}^4 \mapsto \mathbb{R}^3$ is a projection function for the $i$th view, $K_i \in \mathbb{R}^{3 \times 3}$ is the camera matrix that contains the intrinsic parameters that describe a camera’s characteristics (e.g., focal length, offset), and $R_{c,i} \in \mathbb{R}^{3 \times 3}$ and $t_{c,i} \in \mathbb{R}^3$ denote the rotation matrix and translation vector from the world reference system to the camera reference system, respectively.

Let us assume that the world reference system is associated with the first camera for the first viewpoint, with other cameras offsetting the first camera by rotation $R_{c,i,1}$ and translation $t_{c,i,1}$. The projected $m$ views thus become

$$V_1 = P_1X^T = K_1 [I \mid 0_{3 \times 1}] X^T,$$

$$V_i = P_iX^T = K_i [R_{c,i,1} \mid t_{c,i,1}] X^T, \quad i = 2, \ldots, m,$$

(4)

where $P_i \in \mathbb{R}^{3 \times 4}$ is the camera projection matrix and $I$ is a $3 \times 3$ identity matrix.

Subsequent to the transformation on $X$, the $m$ views will transform accordingly, leading to $\hat{\mathcal{V}} = \{\hat{V}_1, \ldots, \hat{V}_m\}$,

$$\hat{V}_i = p_i(\hat{X}) = p_i(t(X)) = K_i[R_{c,i} \mid t_{c,i}]\hat{X}^T, \quad i = 1, \ldots, m.$$  

(5)

Though $V_i$ and $\hat{V}_i$ are projected along the same viewpoint $i$ (i.e., the same camera setup), they are projections of the original 3D object and its transformed counterpart, thus demonstrating different perspectives of the same 3D object.

In particular, as will be proved next, when we perform 3D transformations on the 3D object, the 2D views projected from the 3D object via fixed viewpoints (e.g., $V_i$ and $\hat{V}_i$) will transform equivariantly, which is coined the multi-view transformation equivariance.

3.1.2 Multi-view Transformation Equivariance. Before deriving our multi-view transformation equivariance, we first provide the definition of 2D transformation equivariance based on Reference [51].

Definition 1. Given a 2D image $I$ and a 2D transformation $t_I$ applied to the image, a function $E(\cdot)$ that encodes the representation of each image is defined to be 2D transformation equivariant if it satisfies

$$E(t_I(I)) = \rho(t_I)E(I),$$

(6)

where $\rho(t_I)$ is a homomorphism of transformation $t_I$ in the representation space.
Based on the above definition, we next prove the multi-view transformation equivariance under a 3D transformation of the 3D object, which lays the foundation of the proposed formulation.

**Theorem 1.** Given a 3D object $X$, a projection function $p : \mathbb{R}^3 \mapsto \mathbb{R}^2$ that maps a 3D object to a 2D view, a 3D transformation $t$ applied to the 3D object, and a 2D transformation equivariant function $E(\cdot)$ that encodes the representation of each projected view, there exists a homomorphism transformation $\rho(t)$ of transformation $t$ in the representation space, such that function $E(\cdot)$ satisfies multi-view transformation equivariance, i.e.,

$$E(p(t(X))) = \rho(t)E(p(X)).$$  

(7)

To provide proof for Theorem 1, we first prove the following Lemma:

**Lemma 1.** Given a 3D object $X$ and a 3D transformation $t$, as well as a projection function $p : \mathbb{R}^3 \mapsto \mathbb{R}^2$ that maps the 3D object to a 2D view, there exists a transformation $t_p$ such that

$$p(t(X)) = t_p(p(X)).$$  

(8)

**Proof.** Given the intrinsic camera matrix $K$ and the relative camera transformation $R_c$ and $t_c$, Equation (8) can be expanded by the mathematical descriptions in Equations (2) and (3),

$$K[R_c \mid t_c]AX^T = t_pK[R_c \mid t_c]X^T.$$  

(9)

When the following equation is satisfied, Equation (9) holds:

$$PA = t_pP,$$  

(10)

where $P = K[R_c \mid t_c] \in \mathbb{R}^{3 \times 4}$ is the projection matrix. Next, we prove that there exists a solution of $t_p$ to ensure Equation (10).

We first rewrite Equation (10) by multiplying $P^T$ on both sides of the equation, i.e.,

$$PAP^T = t_pPP^T.$$  

(11)

Since the projection matrix $P$ is full rank and thus $PP^T$ is invertible, the transformation $t_p$ is solvable as

$$t_p = PAP^T(PP^T)^{-1}.$$  

(12)

Hence, there exists such a transformation $t_p$ to ensure Lemma 1. □

Having proved Lemma 1, we next provide the proof of Theorem 1.

**Proof.** We first apply a 2D transformation equivariant function $E(\cdot)$ on both sides of Equation (8),

$$E(p(t(X))) = E(t_p(p(X))).$$  

(13)

where $p(X)$ is a projected 2D view, i.e., a 2D image. Thus, leveraging Definition 1, if function $E(\cdot)$ is 2D transformation equivariant, then there exists a homomorphism transformation $\rho_p(t_p)$, such that

$$E(t_p(p(X))) = \rho_p(t_p)E(p(X)).$$  

(14)

Combining Equations (13) and (14), we have

$$E(p(t(X))) = \rho_p(t_p)E(p(X)).$$  

(15)

Since transformation $t_p$ is a function of the 3D transformation $t$ as derived in Equation (12), there exists a function $\rho(t) = \rho_p(t_p)$, which leads to Equation (7) in Theorem 1. Hence, the multi-view transformation equivariance is proved. □
3.1.3 The Formulation. We aim to learn such a function \(E(\cdot)\) that satisfies the equivariance in Theorem 1. Specifically, we train a shared representation learning module \(E(\cdot)\) that extracts the feature representations of multiple views. As stated in Theorem 1, when the feature representations \(E(p(t(X)))\) and \(E(p(X))\) are known, \(p(t)\) can be solved from them. This inspired us to design a decoding module \(D(\cdot)\) to accurately estimate the 3D transformation \(t\) from the feature representations of views before and after transformations, i.e., \(E(p(t(X)))\) and \(E(p(X))\), which enforces the representation learning to satisfy the definition of multi-view transformation equivariance in Theorem 1.

In the setting of self-supervised learning, we formulate MV-TER as a regularizer along with the (semi-)supervised loss of a specific task to train the entire network. Given a neural network with learnable parameters \(\Theta\), the network is trained end-to-end by minimizing the weighted sum of two loss functions:

- The loss of a specific task \(\ell_{\text{task}}\), e.g., a cross-entropy loss in 3D object classification:
  \[
  \ell_{\text{task}} = -\mathbb{E}_{X \sim M} \sum_{c=1}^{C} y^{(c)} \log \hat{y}^{(c)},
  \]  
where \(C\) is the number of classes, \(y = \{y^{(1)}, \ldots, y^{(C)}\}\) is the ground-truth class label, and \(\hat{y} = \{\hat{y}^{(1)}, \ldots, \hat{y}^{(C)}\}\) is the predicted class label.
- The MV-TER loss that is the expectation of estimation error \(\ell_X(t, \hat{t})\) over each sample \(X\) given a distribution of 3D objects \(M\) and each transformation \(t \sim T\). In our experiments, we adopt the mean squared error to implement this loss,
  \[
  \ell_X(t, \hat{t}) = \mathbb{E}_{t \sim T} \mathbb{E}_{X \sim M} \|t - \hat{t}\|_2^2.
  \]  
Thus, the overall loss function becomes
  \[
  \mathcal{L} = \ell_{\text{task}} + \lambda \mathbb{E}_{t \sim T} \mathbb{E}_{X \sim M} \ell_X(t, \hat{t}),
  \]  
where \(\lambda\) is a weighting parameter to strike a balance between the loss of a specific task and the MV-TER loss. Here, the loss \(\ell_{\text{task}}\) can be taken over all the data labels (fully supervised) or partial labels (semi-supervised). In Equation (18), \(\hat{t}\) is decoded as a function of \(V\) and \(\bar{V}\) in multiple views as defined in Equations (3) and (5), and we will present two schemes to decode \(\hat{t}\) in the next subsection.

3.1.4 Two Transformation Decoding Schemes. We propose two schemes to decode the transformation \(t\) in Equation (18) from the feature representations of multiple views \(E(V_i)\) and \(E(\bar{V}_i)\), \(i = 1, \ldots, m\).

**Fusion Scheme.** The first scheme is to decode from fused representations of multiple views (before and after transformations). Suppose the neural network extracts features of \(V_i\) and \(\bar{V}_i\) from a representation learning module \(E(\cdot)\) and estimates the 3D transformation from both features via a transformation decoding module \(D(\cdot)\), then we have
  \[
  \hat{t} = D \left[ F \left( E(V_1), \ldots, E(V_m) \right), F \left( E(\bar{V}_1), \ldots, E(\bar{V}_m) \right) \right].
  \]  
where \(F(\cdot)\) is a function of feature fusion. The fused feature of multiple views essentially represents the 3D object representation.

**Average Scheme.** In the second decoding scheme, we estimate the transformation \(\hat{t}\) from each view before and after transformation and then take the average of the estimates. The idea is each view captures the projected 3D structures under a transformation. This essentially models a 3D object from different perspectives from which the underlying 3D transformation can be revealed.
By averaging the estimated transformations across multiple views, a reasonable estimation of the 3D transformation can be made.

This actually pushes the model to learn a good 3D representation from individual 2D views and leads to an estimation $\hat{t}_i$ from the $i$th view:

$$\hat{t}_i = D\left[ E(V_i), E(\bar{V}_i) \right], \ i = 1, \ldots, m.$$  \hfill (20)

The final decoded 3D transformation is taken as the expectation of $\hat{t}_i$'s:

$$\hat{t} = \frac{1}{m} \sum_{i=1}^{m} \hat{t}_i.$$  \hfill (21)

Hence, we update the parameters $\Theta$ in the representation learning module $E(\cdot)$ and the transformation decoding module $D(\cdot)$ iteratively by backward propagation of the regularized loss in Equation (18).

Please note that, since we employ a transformation decoding module $D(\cdot)$ to map the feature representations to the same space as the ground-truth transformation $t$, i.e., $t$ and $\bar{t}$ have the same dimensions, we can directly optimize the mean squared error in (17). Interestingly, the second decoding scheme can reach an even better performance than the first scheme in experiments. This should not be surprising, since our ultimate goal is to enable multi-view learning by fusing the representations of individual 2D views to reveal the target 3D objects. The second scheme follows this motivation by pushing each view to encode as much information as possible about the 3D transformation, as implied by the multi-view learning.

3.1.5 Discussions on the Proposed MV-TER. We provide some analysis and discussion on how the MV-TER loss helps to learn transformation equivariant representations, as well as the difference between data augmentation and MV-TER.

**The MV-TER Loss.** We minimize the MV-TER loss to enforce the property of the multi-view transformation equivariance as in Equation (7). Specifically, we implement the definition by decoding the 3D transformation parameters $t$ from the feature representations of multiple views before and after the transformation, i.e., $E(V_i)$ and $E(\bar{V}_i)$. The proposed MV-TER loss measures the estimation accuracy of the 3D transformation, thus minimizing the MV-TER loss leads to the learning of transformation equivariant representations.

**Difference between Data Augmentation and MV-TER.** Compared with data augmentation where the transformed samples are used as additional training samples, the proposed MV-TER differs in the following three aspects:

- Data augmentation needs to mark different augmented versions of a sample with the same label, while the proposed self-supervised MV-TER does not require the label information of transformed samples.
- Data augmentation needs to ensure the “class invariance” of samples, which limits the types of data augmentation (e.g., rotation-based data augmentation cannot be used in handwriting digit recognition, because “6” would be rotated to “9”). In comparison, the proposed MV-TER aims at modeling transformation equivariance, which admits a larger family of 3D transformations used in the MV-TER loss that measures the estimation accuracy of the applied transformations.
- The property of transformation equivariant representations is not guaranteed by simply adopting data augmentation, as the transformation equivariance and class invariance are two different concepts in unsupervised and supervised settings, respectively.
3.2 The Model

Given a 3D object $X$, we randomly draw a transformation $t \sim \mathcal{T}$ and apply it to $X$ to obtain a transformed $\tilde{X}$. Then, we have $m$ views $\mathcal{V} = \{V_1, \ldots, V_m\}$ by projecting $X$ to 2D views. Accordingly, the views after the 3D transformation are $\tilde{\mathcal{V}} = \{\tilde{V}_1, \ldots, \tilde{V}_m\}$.

To learn the applied 3D transformation $t$, we design an end-to-end architecture as illustrated in Figures 1(a) and 1(b) for the fusion and average decoding scheme, respectively. We choose existing CNN models as the representation learning module $E(\cdot)$ (e.g., AlexNet [34], GoogLeNet [59]), which extracts the representation of each view separately. The learned feature representations are then fed into a fusion module and a transformation decoding module $D(\cdot)$, respectively. The fusion module is to fuse the features of multiple views as the overall 3D object representation, e.g., by a view-wise max-pooling layer [57] or group pooling layer [11]. The fused feature will serve as the general descriptor of the 3D object for the subsequent downstream learning tasks (e.g., classification and retrieval). The transformation decoding module $D(\cdot)$ is to estimate the 3D transformation parameters from the feature representations of views before and after transformation. Next, we will discuss the representation learning module and transformation decoding module in detail.

3.2.1 Representation Learning Module. The representation learning module $E(\cdot)$ takes the original 2D views $\mathcal{V} = \{V_1, \ldots, V_m\}$ and their transformed counterparts $\tilde{\mathcal{V}} = \{\tilde{V}_1, \ldots, \tilde{V}_m\}$ as the
input, where $V_i \in \mathbb{R}^{H \times W \times 3}$ and $H$ and $W$ represent the height and width of the 2D view, respectively. $E(\cdot)$ learns feature representations of $V$ and $\hat{V}$ through a Siamese encoder network with shared weights, i.e., $E : V_i \mapsto E(V_i), E(\hat{V}_i) \in \mathbb{R}^C$, where $C$ is the channels of the output features.

Specifically, we employ the feature learning layers of a pre-trained CNN model (e.g., AlexNet [34], GoogLeNet [59]) as the backbone. For instance, we remove the last linear layer from GoogLeNet as the feature learning layers to extract the feature representations of each view. Then, we obtain the features of each view before and after transformation, i.e., \{E(V_i), E(\hat{V}_i)\}, $i = 1, \ldots, m$.

3.2.2 Transformation Decoding Module. To estimate the 3D transformation $t$, we concatenate extracted features of multiple views before and after transformation at feature channels, which are then fed into the transformation decoding module. The module consists of one linear layer to aggregate the representations of multiple views for the prediction of the 3D transformation. As discussed in Section 3.1.4, we have two strategies for decoding the transformation parameters.

- **Fusion Scheme.** We first employ a view-wise pooling layer $F(\cdot)$ (e.g., max-pooling layer [57] or group pooling layer [11]) as in Equation (19) to acquire the fused feature representations of $m$ views, i.e., $F(V_1, \ldots, V_m) \in \mathbb{R}^C$. Then, we concatenate the feature vector of the fused representations from the original and transformed multiple views to acquire a $2C$-dimensional feature vector, which will be fed into a linear layer to estimate the transformation $\hat{t}$.

- **Average Scheme.** Having obtained the feature vectors \{E(V_i), E(\hat{V}_i)\} of each view pair of the original and transformed 3D object, we first concatenate the two representations for each view pair to acquire $m$ $2C$-dimensional feature vectors, which will be fed into the linear layer separately to estimate the transformation $t_i$, $i = 1, \ldots, m$. The final transformation $\hat{t}$ is calculated by averaging the $m$ transformations, as in Equation (21).

Based on the loss in Equation (18), the entire network is trained by minimizing the mean squared error between the ground truth and estimated transformation parameters $t$ and $\hat{t}$ and the task loss (e.g., the cross-entropy loss).

4 EXPERIMENTS

In this section, we evaluate the proposed MV-TER model on two representative downstream tasks: 3D object classification and retrieval. In particular, we apply the MV-TER to supervised 3D object classification in Section 4.2 and retrieval in Section 4.3. Then, we evaluate the proposed model in semi-supervised and unsupervised fashion in Sections 4.4 and 4.5, respectively. Ablation studies are conducted in Section 4.6 to demonstrate the effectiveness of the MV-TER model. Further, we show the generalization performance on both synthetic and real-world datasets in Section 4.7 and evaluate the 3D transformation estimation in Section 4.8.

4.1 Datasets and Preprocessing

We employ four datasets to evaluate the classification performance of the proposed MV-TER model, including two synthetic datasets ModelNet40 [69] and ShapeNetCore55 [4] and two real-world multi-view datasets RGB-D [35] and ETH-80 [37].

ModelNet40 [69]. This dataset contains 12,311 CAD models from 40 categories. We follow the standard training and testing split settings, i.e., 9,843 models are used for training and 2,468 models are for testing.

ShapeNetCore55 [4]. This dataset contains 51,162 CAD models categorized into 55 classes. The training, validation, and test sets consist of 35,764, 5,133, and 10,265 objects, respectively.
RGB-D [35]. This dataset is a real-world multi-view dataset containing both RGB and depth images of 300 objects from 51 categories. We sample 24 views for each object instance for training and testing. We perform 10-fold cross validation to report the average accuracies as in Reference [35].

ETH-80 [37]. This dataset contains 80 real-world models from eight categories. For each category, there are 10 object instances and 41 views for each object instance captured from different viewpoints. We employ the leave-one-object-out cross-validation strategy as suggested in Reference [37] to report the classification accuracies.

To acquire projected 2D views of the synthetic 3D object datasets ModelNet40 and ShapeNetCore55, we follow the experimental settings of MVCNN [57] to render multiple views of each 3D object. Here, 12 virtual cameras are employed to capture views with an interval angle of 30 degrees. We acquire the 12 views by fixing the camera setup for all 3D object instances in the whole dataset, and the viewpoints of the 12 views are the same across objects. Next, we employ rotation as our 3D transformations on objects and perform a random rotation with three parameters all in the range $[-\pi, \pi]$ on the entire 3D object. We also render the views of the transformed 3D object using the same settings as the original 3D object. The rendered multiple views before and after 3D transformations are taken as the input to our method.

4.2 3D Object Classification

In this task, we employ the MV-TER as a self-supervisory regularizer to two competitive multi-view-based 3D object classification methods: MVCNN [57] and GVCNN [11], which are referred to as MV-TER (MVCNN) and MV-TER (GVCNN). Further, we implement two transformation decoding schemes as discussed in Section 3.1.4, including the fusion scheme and average scheme. Then, our model has four variants, as presented in Table 1.

Implementation Details: We deploy GoogLeNet [59] as our backbone CNN as in GVCNN [11]. The backbone GoogLeNet is pre-trained on ImageNet1K dataset. We remove the last linear layer as the Siamese representation learning module to extract features for each view. Subsequent to the representation learning module, we employ one linear layer as the transformation decoding module. The output feature representations of the Siamese network first go through a channel-wise concatenation, which are then fed into the transformation decoding module to estimate the transformation parameters. The entire network is trained via the SGD optimizer with a batch size of 24. The momentum and weight decay rate are set to 0.9 and $10^{-4}$, respectively. The initial learning rate is 0.001, which then decays by a factor of 0.5 for every 10 epochs. The weighting parameter $\lambda$ in Equation (18) is set to 0.5. Also note that MVCNN in Table 1 has two variants with different backbones: MVCNN [57] and MVCNN (GoogLeNet) [11]. MVCNN [57] uses the VGG-M [5] as the backbone, while MVCNN (GoogLeNet) implemented in Reference [11] employs the GoogLeNet [59]. In addition, RotationNet [30] and View-GCN [67] are set up with 12 views taken by the default camera system for fair comparison. Please note that the methods listed in Table 1 are all fully supervised, where the labels of all the training data are used as the supervision signals.

Experimental Results: As listed in Table 1, the MV-TER (MVCNN), average and MV-TER (GVCNN), average achieve classification accuracy of 95.5% and 97.0%, respectively, which outperform the state-of-the-art View-GCN [67]. Also, the MV-TER (MVCNN), average outperforms its baseline MVCNN (GoogLeNet) by 3.3%, while MV-TER (GVCNN), average outperforms the baseline GVCNN (GoogLeNet) by 4.4%, which demonstrates the effectiveness of our proposed MV-TER as a self-supervisory regularizer.

4.3 3D Object Retrieval

In this task, we directly employ the fused feature representations of MV-TER (MVCNN) and MV-TER (GVCNN) as the 3D object descriptor for retrieval. We denote $F_X$ and $F_Y$ as the 3D object
Table 1. 3D Object Classification and Retrieval Results on ModelNet40 Dataset

| Methods                  | Backbone | Training Configuration | Modality | Classification (Accuracy) | Retrieval (mAP) |
|--------------------------|----------|------------------------|----------|---------------------------|-----------------|
| 3D ShapeNets [69]        | -        | ModelNet40              | voxels   | 77.3                      | 49.2            |
| VoxNet [45]              | -        | ModelNet40              | voxels   | 83.0                      | -               |
| SubvolumeSup [48]        | -        | ModelNet40              | voxels   | 89.2                      | -               |
| PointNet [47]            | -        | ModelNet40              | points   | 89.2                      | -               |
| PointNet++ [49]          | -        | ModelNet40              | points   | 91.9                      | -               |
| KD-Networks [33]         | -        | ModelNet40              | points   | 91.8                      | -               |
| MVCNN [57]               | VGG-M    | ImageNet1K              | ModelNet40 | 12 views | 89.9            | 70.1            |
| MVCNN, metric [57]       | VGG-M    | ImageNet1K              | ModelNet40 | 12 views | 89.5            | 80.2            |
| MVCNN, multi-resolution [48] | AlexNet | ImageNet1K              | ModelNet40 | 20 views | 93.8            | -               |
| RotationNet [30]         | VGG-M    | ImageNet1K              | ModelNet40 | 12 views | 90.7            | -               |
| SeqViews2SeqLabels [20]  | VGG-19   | ImageNet1K              | ModelNet40 | 12 views | 93.4            | 89.1            |
| 3D2SeqViews [19]         | VGG-19   | ImageNet1K              | ModelNet40 | 12 views | 93.4            | 90.8            |
| Relation Network [72]    | VGG-M    | ImageNet1K              | ModelNet40 | 12 views | 94.3            | 86.9            |
| MLVCNN, Center Loss [29] | ResNet-18| ImageNet1K              | ModelNet40 | 36 views | 94.2            | 92.8            |
| View-GCN [67]            | ResNet-18| ImageNet1K              | ModelNet40 | 12 views | 96.2            | -               |
| MVLADN [73]              | VGG-M    | ImageNet1K              | ModelNet40 | 12 views | 93.0            | -               |
| CAR-Net [70]             | ResNet-18| ImageNet1K              | ModelNet40 | 12 views | 95.2            | 91.3            |
| MVCNN [11]               | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 92.2            | 74.1            |
| MVCNN, metric [11]       | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 92.2            | 83.0            |
| MV-TER (MVCNN), average | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 95.5            | 83.0            |
| MV-TER (MVCNN), fusion   | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 95.5            | 84.9            |
| MV-TER (MVCNN), average, Center Loss | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 95.1            | 86.6            |
| MV-TER (MVCNN), fusion, Center Loss | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 95.5            | 87.2            |
| GVCNN [51]               | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 92.6            | 81.3            |
| GVCNN, metric [11]       | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 92.6            | 85.7            |
| MV-TER (GVCNN), average | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 97.0            | 88.8            |
| MV-TER (GVCNN), fusion   | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 96.4            | 88.3            |
| MV-TER (GVCNN), average, Center Loss | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 95.7            | 91.5            |
| MV-TER (GVCNN), fusion, Center Loss | GoogLeNet| ImageNet1K              | ModelNet40 | 12 views | 96.3            | 91.1            |

descriptor of two 3D objects X and Y, respectively, and use the Euclidean distance between them for retrieval. The metric is defined as \( \text{dist}(X, Y) = \| F_X - F_Y \|_2 \).

We take the **mean average precision (mAP)** on retrieval as the evaluation metric and present the comparison results in the last column of Table 1. For MVCNN and GVCNN, a low-rank Mahalanobis metric learning [57] is applied to boost the retrieval performance. In comparison, we train our MV-TER model without the low-rank Mahalanobis metric learning, but still achieve better retrieval performance, which validates the superiority of our feature representation learning for 3D objects. Further, we apply Triplet Center Loss [25] to our MV-TER. With Center Loss, our model further achieves an average gain of 3.3% in mAP. As presented in the last column of Table 1, the **MV-TER (GVCNN), average** and **MV-TER (GVCNN), fusion** achieve mAP of 91.5% and 91.1%, respectively, which is comparable to MLVCNN with Center Loss [29] while we only take 12 views as input instead of 36 views. We demonstrate some visual results of 3D object retrieval in Figure 2.

### 4.4 Semi-supervised Learning

In this experiment, we only use a small number of labeled data to train our model to further evaluate the effectiveness of the proposed self-supervisory regularizer MV-TER. We adopt six different label rates in the set \( \{0.01, 0.02, 0.03, 0.04, 0.05, 0.10\} \) to train four models for comparison: MVCNN (AlexNet), GVCNN (AlexNet), **MV-TER (MVCNN), average** and **MV-TER (GVCNN), average**. When training MVCNN (AlexNet) and GVCNN (AlexNet), we only use a small amount of labeled data to minimize the cross-entropy loss for training and then employ all the test data for evaluation. When training **MV-TER (MVCNN), average** and **MV-TER (GVCNN), average**, we adopt all the data (labeled and unlabeled) to predict the 3D transformations without the use of labels and...
then adopt only labeled data to acquire classification scores. That is, we minimize Equation (18) with a small number of labels taken for the classification loss $\ell_{\text{task}}$. In all four models, a pre-trained AlexNet [34] on ImageNet1K dataset is employed as the backbone CNN.

Figure 3 presents the classification accuracy under the six label rates on ModelNet40 dataset. When the label rate is 0.10, we see that the four models achieve comparable results, which benefits from the pre-training of the backbone AlexNet. When the label rate keeps decreasing, the performance of both MVCNN and GVCNN drops quickly, while the MV-TER models are much more robust. Even at the extremely low label rate 0.01, **MV-TER (MVCNN), average** and **MV-TER (GVCNN), average** achieve the classification accuracy of 58.6% and 55.2%, respectively, thus demonstrating the robustness of the proposed MV-TER model.

### 4.5 Unsupervised Learning

To demonstrate the effectiveness of the MV-TER model, we design an unsupervised architecture for the learning of multi-view transformation equivariant representations given multiple views before and after 3D transformations. To validate the proposed model better, we choose the simpler AlexNet as the backbone architecture to minimize the influence of more complex CNN structures (e.g., ResNet) on the results. The MV-TER model is trained on the ModelNet40 dataset in an
unsupervised fashion by directly minimizing the loss $\ell_X(t, \hat{t})$ as defined in Equation (18), without any reliance on labels. We feed the projected multiple views before and after transformations into the AlexNet with shared weights to learn the feature representations, which are then used to decode the 3D transformation with the average decoding scheme via one linear layer. After the unsupervised training, we take the last fully connected layer of the trained AlexNet with weights frozen to extract the 4,096-dimensional feature vector as the representation of each 3D object and train a linear SVM classifier using the “one-vs-rest” strategy to acquire classification scores.

As reported in Table 2, we compare with the fully supervised model that provides the upper bounded performance by training the AlexNet with all labeled data, denoted as “MV-TER (Supervised).” Meanwhile, we employ two variants of AlexNet without or with ImageNet pretraining, denoted as “MV-TER” and “MV-TER (ImageNet),” where the latter is for the comparison with the state-of-the-art method “LCGC (ImageNet)” [40]. As in Table 2, the performance of the MV-TER without ImageNet pretraining even achieves comparable results to the state-of-the-art method LCGC with ImageNet and outperforms LCGC (ImageNet) by 1.38% when using the ImageNet pretrained backbone. Further, the performance of MV-TER closely reaches that of fully supervised learning, which demonstrates that the unsupervised learning of the MV-TER greatly narrows the performance gap to the upper bound.
Table 3. Classification Accuracy (%) and Retrieval mAP (%) on ModelNet40 Dataset with Different Transformation Strategies

| Method                        | Rotation Accuracy | Shearing Accuracy | Rotation+Shearing Accuracy |
|-------------------------------|-------------------|-------------------|----------------------------|
| MV-TER (MVCNN), average      | 95.5              | 83.0              | 94.5                       |
| MV-TER (MVCNN), fusion       | 95.5              | 84.9              | 94.1                       |
| MV-TER (GVCNN), average      | 97.0              | 88.8              | 94.7                       |
| MV-TER (GVCNN), fusion       | 96.4              | 88.3              | 94.7                       |
| mean                          | 96.1              | 86.3              | 94.5                       |

Table 4. Experimental Comparison between GVCNN and the Proposed Model with Different Number of Input Views for 3D Object Classification on ModelNet40 Dataset

| Training Views | Testing Views | GVCNN Accuracy (%) | MV-TER Accuracy (%) |
|----------------|--------------|--------------------|---------------------|
| 8              | 2            | 71.2               | 91.9                |
|                | 4            | 91.1               | 94.6                |
|                | 8            | 93.1               | 95.4                |
|                | 12           | 91.5               | 96.0                |
| 12             | 2            | 76.8               | 84.3                |
|                | 4            | 90.3               | 92.5                |
|                | 8            | 92.1               | 95.8                |
|                | 12           | 92.6               | 97.0                |

4.6 Ablation Studies

4.6.1 On Different 3D Transformations. We first evaluate the effectiveness of different transformation strategies on the ModelNet40 dataset. We apply three types of 3D transformations to the 3D objects, including (1) Rotation: randomly rotate each 3D object with three rotation parameters all in the range \([-\pi, \pi]\); (2) Shearing: randomly shear the x-, y-, z-coordinates of each 3D object with the six parameters of a shearing matrix in the range \([-0.3, 0.3]\); (3) the composite transformation of Rotation and Shearing: We first randomly shear the x-, y-coordinates of each 3D object with the two parameters of a shearing matrix in the range \([-0.3, 0.3]\) and then randomly rotate the object with three rotation parameters in the range \([-\pi, \pi]\). Table 3 presents the classification accuracy and retrieval mAP with the three types of 3D transformation strategies. As we can see, under the 3D rotation and shearing transformations, the proposed model achieves comparable results on average; under the composite transformation of 3D rotation and shearing, our model still achieves reasonable results. This demonstrates that our model is insensitive to different 3D transformations and shows its stability under complex 3D transformations.

4.6.2 On the Number of Views. We quantitatively evaluate the influence of the number of views on the classification task. Specifically, we randomly choose \{8, 12\} views from all the views as the input to train MV-TER (GVCNN), average, respectively, leading to two learned networks. Then, we randomly select \{2, 4, 8, 12\} views from all the testing views to evaluate the classification accuracy of the two networks, respectively, as reported in Table 4. We see that we constantly outperform GVCNN with different numbers of training views and testing views. In particular, when the number of testing views reaches the extreme of two views for multi-view learning, our MV-TER
Table 5. Classification Accuracy (%) of MV-TER on Different Backbones

| Backbone     | MV-TER, average MVCNN | MVCNN | GVCNN | MV-TER, fusion MVCNN | GVCNN | Mean  |
|--------------|------------------------|-------|-------|-----------------------|-------|-------|
| AlexNet      | 94.2                   | 93.5  |       | 94.1                  | 94.6  | 94.1  |
| VGG-M        | 95.0                   | 95.5  |       | 94.2                  | 95.3  | 95.0  |
| VGG-19       | 95.9                   | 96.4  |       | 96.7                  | 96.6  | 96.4  |
| ResNet-18    | 96.0                   | 96.4  |       | 96.1                  | 96.4  | 96.2  |
| GoogLeNet    | 95.5                   | 97.0  |       | 95.5                  | 96.4  | 96.1  |

Table 6. Classification Accuracy (%) of MV-TER with Different Transformation Decoding Module Architectures

| Model Configuration | MV-TER, average MVCNN | MVCNN | GVCNN | MV-TER, fusion MVCNN | GVCNN | Mean  |
|---------------------|------------------------|-------|-------|-----------------------|-------|-------|
| {4096, 3}           | 95.5                   | 97.0  |       | 95.5                  | 96.4  | 96.1  |
| {4096, 1024, 3}     | 95.4                   | 96.8  |       | 95.6                  | 96.5  | 96.1  |
| {4096, 1024, 512, 3}| 95.6                   | 96.7  |       | 95.5                  | 96.6  | 96.1  |
| {4096, 1024, 512, 256, 3}| 95.6 | 97.0   |       | 95.8                  | 96.3  | 96.2  |
| {4096, 1024, 512, 256, 128, 3} | 95.5 | 97.2   |       | 95.7                  | 96.7  | 96.3  |

model is still able to achieve the classification accuracy of 91.9% and 91.2%, which outperforms GVCNN by a large margin.

It is worth noting that though views are projected along fixed viewpoints in the training stage, the proposed model learns the transformation equivariant representations that capture the intrinsic and generalizable representations of multiple views, which is independent of how the views are sampled. This is validated by the experimental results in Table 4: We train the MV-TER on 8 views and test on 12 views where some views are not present in the training data. Results show that the MV-TER achieves the classification accuracy of 96.0%, which is comparable to training and testing on 12 views.

4.6.3 Ablation Study on Different Backbones. We further investigate the performance of MV-TER with five different backbones: AlexNet [34], VGG-M [5], VGG-19 [54], ResNet-18 [24], and GoogLeNet [59]. As presented in Table 5, we can see that the classification accuracies are comparable on average under the five backbones. Moreover, our method even achieves an average accuracy of 94.1% under the simplest AlexNet backbone. This shows that the performance of our proposed method relies little on the backbone. Also, we notice that the proposed method outperforms competitive methods under the same backbone shown in Table 1. This further validates the advantage of our method.

4.6.4 Ablation Study on Different Architectures of Transformation Decoding Module. We further evaluate the classification accuracy with different numbers of linear layers in the transformation decoding module. The number of linear layers ranges from one to five, and the specific number of hidden channels of each layer is specified in the first column of Table 6. As we can see, our proposed MV-TER achieves similar classification accuracy at different numbers of linear layers. Meanwhile, there is a slight improvement in the average classification accuracy as the number of layers increases. This indicates that our model achieves reasonable and robust results with a simple transformation decoding architecture and a smaller number of learnable parameters, which inspires us to design a simple architecture and reduce the number of parameters to improve the computational efficiency.
Table 7. Classification Comparison of MV-TER and Two Baseline Methods under the ShapeNetCore55 Pre-training Strategy

| Method         | Accuracy (%) | Method         | Accuracy (%) |
|----------------|--------------|----------------|--------------|
| **ModelNet40** |              | **RGB-D**      |              |
| MVCNN          | 85.9         | GVCNN          | 87.9         |
| MV-TER, average| 88.7         | MV-TER, average| 91.2         |
| MV-TER, fusion | 89.0         | MV-TER, fusion | 90.3         |
| **ETH-80**     |              |                |              |
| MVCNN          | 74.12 ± 4.22 | GVCNN          | 76.41 ± 4.08 |
| MV-TER, average| 75.75 ± 2.59 | MV-TER, average| 78.56 ± 3.52 |
| MV-TER, fusion | 76.60 ± 4.30 | MV-TER, fusion | 80.20 ± 3.18 |

4.7 Transfer Learning

We further show the generalization performance of the proposed MV-TER under average and fusion schemes. We take the same network architecture and parameter settings as in Section 4.2. In particular, we train the MV-TER model on the ShapeNetCore55 dataset by using all the labeled data and test on other datasets by a linear SVM classifier using the feature representations of dimension 2,048 obtained from the second last fully connected layer of MV-TER. Table 7 reports the classification comparison of MV-TER and two baseline methods on three datasets under the ShapeNetCore55 pre-training strategies. As we can see, the proposed MV-TER with two decoding schemes improves the average classification accuracy by 2.95% and 2.85%, respectively, on the synthetic dataset ModelNet40 under the ShapeNetCore55 pre-training strategy compared with the two baseline methods MVCNN and GVCNN, thus validating the generalizability.

We then evaluate our MV-TER on the real-world multi-view datasets RGB-D and ETH-80. As shown in the middle and bottom parts of Table 7, compared with the two baseline methods MVCNN and GVCNN, the MV-TER improves the average classification accuracy by 2.05% and 2.97%, respectively, on the RGB-D dataset, and 1.88% and 1.25% on the ETH-80 dataset. Also, our classification results are more stable with less variation during the cross-validation in general. This demonstrates the generalizability of the MV-TER to real-world data.

4.8 Evaluation of the 3D Transformation Estimation

Further, to intuitively interpret the estimated 3D transformations from the proposed fusion and average decoding schemes, we visualize the multiple views projected from 3D objects Car and Bowl with the estimated 3D transformations applied. In Figures 4(a) and 4(b), the first, second, and fourth rows demonstrate the projected views from the 3D object with the same 3D transformation: the ground truth, the estimation from the fusion scheme, and the estimation from the average scheme. In the third row, each view is the result of each individually estimated 3D transformation $t_i$ as in Equation (20), i.e., view-wise transformations. Note that each column is rendered under the same viewpoint. We see that our MV-TER model estimates more accurate 3D transformations via the average scheme, which is consistent with the objective results.

5 CONCLUSION

In this article, we propose a novel self-supervised learning framework of Multi-View Transformation Equivariant Representations (MV-TER) via auto-encoding 3D transformations,
Fig. 4. Illustration of multiple views projected from 3D objects in the same posture: (a) Car and (b) Bowl. The four rows of (a) and (b) demonstrate multiple views projected from the 3D object with the following 3D transformations applied, respectively: (1) the ground-truth 3D transformation; (2) the estimated 3D transformation of the fusion decoding scheme; (3) the individually estimated 3D transformations $\hat{t}_i$'s from each view during the average decoding scheme (with $t_i$ applied to the $i$th view); and (4) the finally averaged 3D transformation of the average decoding scheme.

exploiting the equivariant transformations of a 3D object and its projected multiple views. We derive the multi-view transformation equivariance and formulate the MV-TER learning accordingly. Specifically, we perform a 3D transformation on a 3D object, which leads to equivariant transformations in projected multiple views. By decoding the 3D transformation from the fused feature representations of multiple views before and after transformation, the MV-TER enforces the representation learning module to learn intrinsic 3D object representations. Experimental results demonstrate that the proposed MV-TER significantly outperforms the state-of-the-art view-based approaches in 3D object classification and retrieval tasks.

REFERENCES

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning representations and generative models for 3D point clouds. In Proceedings of the International Conference on Machine Learning (ICML). PMLR, 40–49.
2] Matan Atzmon, Haggai Maron, and Yaron Lipman. 2018. Point convolutional neural networks by extension operators. ACM Trans. Graph. 37, 4 (July 2018), 1–12.

3] Philip Bachman, R. Devon Hjelm, and William Buchwalter. 2019. Learning representations by maximizing mutual information across views. In Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS). 15509–15519.

4] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su et al. 2015. ShapeNet: An information-rich 3D model repository. arXiv preprint arXiv:1512.03012 (2015).

5] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the devil in the details: Delving deep into convolutional nets. In Proceedings of British Machine Vision Conference (BMVC).

6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In Proceedings of the International Conference on Machine Learning (ICML). PMLR, 1597–1607.

7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. 2020. Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020).

8] Taco Cohen and Max Welling. 2016. Group equivariant convolutional networks. In Proceedings of the International Conference on Machine Learning (ICML). 2990–2999.

9] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. 2016. Exploiting cyclic symmetry in convolutional neural networks. In Proceedings of the International Conference on Machine Learning (ICML). 1899–1898.

10] Sander Dieleman, Kyle W. Willett, and Joni Dambre. 2015. Rotation-invariant convolutional neural networks for galaxy morphology prediction. Mon. Not. Roy. Astronom. Soc. 450, 2 (2015), 1441–1459.

11] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. 2018. GVCNN: Group-view convolutional neural networks for 3D shape recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 264–272.

12] Matheus Gadelha, Rui Wang, and Subhransu Maji. 2018. Multiresolution tree networks for 3D point cloud processing. In Proceedings of the European Conference on Computer Vision (ECCV). 103–118.

13] Xiang Gao, Wei Hu, and Guo-Jun Qi. 2020. GraphTER: Unsupervised learning of graph transformation equivariant representations via auto-encoding node-wise transformations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

14] Zan Gao, Yiming Li, and Shaohua Wan. 2020. Exploring deep learning for view-based 3D model retrieval. ACM Trans. Multim. Comput. Commun. Appl. 16, 1 (2020), 1–21.

15] Zan Gao, Yan Zhang, Hua Zhang, Weili Guan, Dong Feng, and Shengyong Chen. 2021. Multi-level view associative convolution network for view-based 3D model retrieval. IEEE Trans. Circ. Syst. Vid. Technol. 32, 4 (2021), 2264–2278.

16] Robert Gens and Pedro M. Domingos. 2014. Deep symmetry networks. In Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS). 2537–2545.

17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. Adv. Neural Inf. Process. Syst. 27 (2014).

18] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2020. Deep learning for 3D point clouds: A survey. IEEE Trans. Pattern Anal. Mach. Intell. (2020).

19] Zhizhong Han, Honglei Lu, Zhenbao Liu, Chi-Man Vong, Yu-Shen Liu, Matthias Zwicker, Junwei Han, and C. L. Philip Chen. 2019. 3D2SeqViews: Aggregating sequential views for 3D global feature learning by CNN with hierarchical attention aggregation. IEEE Trans. Image Process. 28, 8 (2019), 3986–3999.

20] Zhizhong Han, Mingyang Shang, Zhenbao Liu, Chi-Man Vong, Yu-Shen Liu, Matthias Zwicker, Junwei Han, and C. L. Philip Chen. 2018. SeqViews2SeqLabels: Learning 3D global features via aggregating sequential views by RNN with attention. IEEE Trans. Image Process. 28, 2 (2018), 658–672.

21] Zhizhong Han, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. 2019. Multi-angle point cloud-VAE: Unsupervised feature learning for 3D point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 10441–10450.

22] Zhizhong Han, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. 2021. Hierarchical view predictor: Unsupervised 3D global feature learning through hierarchical prediction among unordered views. In Proceedings of the 29th ACM International Conference on Multimedia. 3862–3871.

23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 9729–9738.

24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 770–778.
[25] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. 2018. Triplet-center loss for multi-view 3D object retrieval. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1945–1954.

[26] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming auto-encoders. In Proceedings of the International Conference on Artificial Neural Networks (ICANN). Springer, 44–51.

[27] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In Proceedings of the International Conference on Learning Representations (ICLR).

[28] Qianjiang Hu, Xiaowei Wang, Wei Hu, and Guo-Jun Qi. 2021. AdCo: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 1074–1083.

[29] Jianwen Jiang, Di Bao, Ziqiang Chen, Xibin Zhao, and Yue Gao. 2019. MLVCNN: Multi-loop-view convolutional neural network for 3D shape retrieval. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). 8515–8520.

[30] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. 2018. RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 5010–5019.

[31] Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114 (2013).

[32] L. J. Kivinen and Christopher K. I. Williams. 2011. Transformation equivariant Boltzmann machines. In Proceedings of the International Conference on Artificial Neural Networks (ICANN). Springer, 1–9.

[33] Roman Klokov and Victor Lempitsky. 2017. Escape from: Deep Kd-networks for the recognition of 3D point cloud models. In Proceedings of the IEEE International Conference on Computer Vision (CVPR). 863–872.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS). 1097–1105.

[35] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. 2011. A large-scale hierarchical multi-view RGB-D object dataset. In Proceedings of the IEEE International Conference on Robotics and Automation. IEEE, 1817–1824.

[36] Truc Le and Ye Duan. 2018. PointGrid: A deep network for 3D shape understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 9204–9214.

[37] Bastian Leibe and Bernt Schiele. 2003. Analyzing appearance and contour based methods for object categorization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, II–409.

[38] Karel Lenc and Andrea Vedaldi. 2015. Understanding image representations by measuring their equivariance and equivalence. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 991–999.

[39] Jan Eric Lenssen, Matthias Fey, and Pascal Libuschewski. 2018. Group equivariant capsule networks. In Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS). 8844–8853.

[40] Wenhui Li, Zhenlan Zhao, An-An Liu, Zan Gao, Chenggang Yan, Zhenhong Mao, Haipeng Chen, and Weizhi Nie. 2022. Joint local correlation and global contextual information for unsupervised 3D model retrieval and classification. IEEE Trans. Circ. Syst. Vid. Technol. 32, 5 (2022), 3265–3278.

[41] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution on x-transformed points. In Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS). 820–830.

[42] Shikun Liu, Lee Giles, and Alexander Ororbia. 2018. Learning a hierarchical latent-variable model of 3D shapes. In Proceedings of the International Conference on 3D Vision (3DV). IEEE, 542–551.

[43] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. IEEE Trans. Knowl. Data Eng. (2021).

[44] Yongchong Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. 2019. Relation-shape convolutional neural network for point cloud analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 8895–8904.

[45] Daniel Maturana and Sebastian Scherer. 2015. VoxNet: A 3D convolutional neural network for real-time object recognition. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 922–928.

[46] Omid Poursaeed, Tianxing Jiang, Han Qiao, Nayun Xu, and Vladimir G. Kim. 2020. Self-supervised learning of point clouds via orientation estimation. In Proceedings of the International Conference on 3D Vision (3DV). IEEE, 1018–1028.

[47] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep learning on point sets for 3D object classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 652–660.

[48] Charles R. Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. 2016. Volumetric and multi-view CNNs for object classification on 3D data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 5648–5656.

[49] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS). 5099–5108.
[50] Guo-Jun Qi, Liheng Zhang, Chang Wen Chen, and Qi Tian. 2019. AVT: Unsupervised learning of transformation equivariant representations by autoencoding variational transformations. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

[51] Guo-Jun Qi, Liheng Zhang, Feng Lin, and Xiao Wang. 2020. Learning generalized transformation equivariant representations via autoencoding transformations. IEEE Tr. Pattern Anal. Mach. Intell. (2020).

[52] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. 2017. OctNet: Learning deep 3D representations at high resolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 3577–3586.

[53] Uwe Schmidt and Stefan Roth. 2012. Learning rotation-aware features: From invariant priors to equivariant descriptors. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2050–2057.

[54] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations (ICLR).

[55] Henrik Skibbe. 2013. Spherical Tensor Algebra for Biomedical Image Analysis. Ph. D. Dissertation. Verlag nicht ermittelbar.

[56] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.

[57] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. 2017. OctNet: Learning deep 3D representations at high resolutions. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 945–953.

[58] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. 2020. PointGrow: Autoregressively learned point cloud generation with self-attention. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 61–70.

[59] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.

[60] Guo-Jun Qi, Liheng Zhang, Feng Lin, and Xiao Wang. 2020. Learning generalized transformation equivariant representations via autoencoding transformations. IEEE Tr. Pattern Anal. Mach. Intell. (2020).

[61] Aaron VanOord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixelrecurrent neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[62] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3D shape recognition. In Proceedings of the IEEE International Conference on Computer Vision (CVPR). 945–953.

[63] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In Proceedings of the International Conference on Machine Learning (ICML). 1747–1756.

[64] Aaron vanOord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In Proceedings of the 26th ACM International Conference on Multimedia. 746–754.

[65] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In Proceedings of the International Conference on Machine Learning (ICML). 1747–1756.

[66] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.

[67] Xin Wei, Ruixuan Yu, and Jian Sun. 2020. View-GCN: View-based graph convolutional network for 3D shape analysis. In Proceedings of the International Conference on Machine Learning (ICML). 1747–1756.

[68] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.

[69] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.

[70] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.

[71] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.

[72] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.

[73] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.

[74] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.
[75] Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. 2019. AET vs. AED: Unsupervised representation learning by auto-encoding transformations rather than data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2547–2555.

[76] Yingxue Zhang and Michael Rabbat. 2018. A graph-CNN for 3D point cloud classification. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 6279–6283.

[77] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 2019. 3D point capsule networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 1009–1018.

Received 19 December 2022; revised 1 April 2023; accepted 9 May 2023