Sparse Steerable Convolutions: An Efficient Learning of SE(3)-Equivariant Features for Estimation and Tracking of Object Poses in 3D Space

Anonymous Author(s)
Affiliation
Address
email

Abstract

As a basic component of SE(3)-equivariant deep feature learning, steerable convolution has recently demonstrated its advantages for 3D semantic analysis. The advantages are, however, brought by expensive computations on dense, volumetric data, which prevent its practical use for efficient processing of 3D data that are inherently sparse. In this paper, we propose a novel design of *Sparse Steerable Convolution (SS-Conv)* to address the shortcoming; SS-Conv greatly accelerates steerable convolution with sparse tensors, while strictly preserving the property of SE(3)-equivariance. Based on SS-Conv, we propose a general pipeline for precise estimation of object poses, wherein a key design is a Feature-Steering module that takes the full advantage of SE(3)-equivariance and is able to conduct an efficient pose refinement. To verify our designs, we conduct thorough experiments on three tasks of 3D object semantic analysis, including instance-level 6D pose estimation, category-level 6D pose and size estimation, and category-level 6D pose tracking. Our proposed pipeline based on SS-Conv outperforms existing methods on almost all the metrics evaluated by the three tasks. Ablation studies also show the superiority of our SS-Conv over alternative convolutions in terms of both accuracy and efficiency. We will make the implementation of SS-Conv publicly available.

1 Introduction

SE(3)-equivariant deep networks [19, 24, 7] have shown the promise recently in some tasks of 3D semantic analysis, among which 3D Steerable CNN [24] is a representative one. 3D Steerable CNNs employ steerable convolutions (termed as *ST-Conv*) to learn pose-equivariant features in a layer-wise manner, thus preserving the pose information of the 3D input. Intuitively speaking, for a layer of ST-Conv, any SE(3) transformation $(\boldsymbol{r}, \boldsymbol{t})$ applied to its 3D input would induce a pose-synchronized transformation to its output features, where $\boldsymbol{r} \in$ SO(3) stands for a rotation and $\boldsymbol{t} \in \mathbb{R}^3$ for a translation. Fig. 1 gives an illustration where given an SE(3) transformation of the input, the locations at which feature vectors are defined are rigidly transformed with respect to $(\boldsymbol{r}, \boldsymbol{t})$, and the feature vectors themselves are also rotated by $\rho(\boldsymbol{r})$ ($\rho(\boldsymbol{r})$ is an representation of rotation $\boldsymbol{r}$). This property of SE(3)-equivariance enables the steerability of feature space. For example, without transforming the input, SE(3) transformation can be directly realized by steering in the feature space. To produce steerable features, ST-Conv confines its feature domain to regular grids of 3D volumetric data; it can thus be conveniently supported by 3D convolution routines. This compatibility with 3D convolutions eases the implementation of ST-Conv, but at the sacrifice of efficiently processing 3D data (e.g., point clouds) that are typically irregular and sparse; consequently, ST-Conv is still less widely used in broader areas of 3D semantic analysis.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

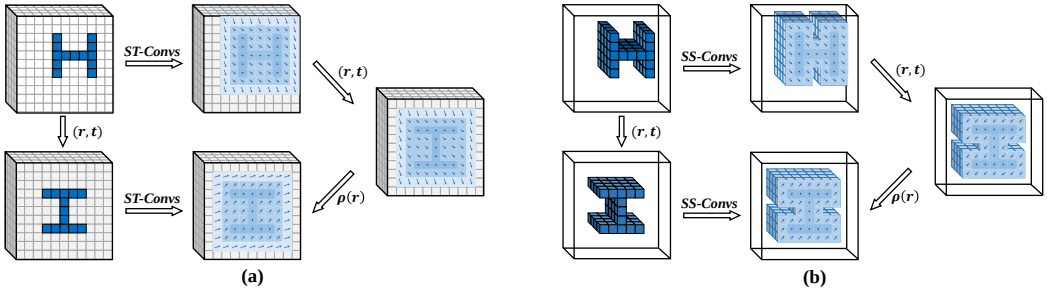


Figure 1: An illustration of SE(3)-equivariance achieved by (a) STeerable Convolution (ST-Conv), and (b) our Sparse Steerable Convolution (SS-Conv), where arrows defined on the 3D fields denote vector-formed, oriented features. Best view in the electronic version.

In this paper, we propose a novel design of *Sparse Steerable Convolution (SS-Conv)* to address the aforementioned shortcoming faced by ST-Conv. SS-Conv can greatly accelerate steerable convolutions with sparse tensors, while strictly preserving the SE(3)-equivariance in feature learning; Fig. 1(b) gives the illustration. To implement SS-Conv, we construct convolutional kernels as linear combinations of spherical harmonics, which satisfy the rotation-steering constraint of SE(3)-equivariant convolutions [24], and implement the convolution as matrix-matrix multiply-add operations on GPUs only at active grids, which are recorded along with their features as sparse tensors.

Although SE(3)-equivariant feature learning is widely used in 3D object recognition, its potentials for other tasks of 3D semantic analysis have not been well explored yet. In this work, we make the attempt to apply our proposed SS-Conv to object pose estimation in 3D space. To this end, we propose a general pipeline based on SS-Conv, which stacks layers of SS-Conv as the backbone, and decodes object poses directly from the learned SE(3)-equivariant features. A novel Feature-Steering module is also designed into the pipeline to support iterative pose refinement, by taking advantage of the steerability of the learned features. We conduct thorough experiments on three tasks of pose-related, 3D object semantic analysis, including instance-level 6D pose estimation, category-level 6D pose and size estimation, and category-level 6D pose tracking. Our proposed pipeline based on SS-Conv outperforms existing methods on almost all the metrics evaluated by the three tasks; the gaps are clearer in the regimes of high-precision pose estimation. Ablation studies also show the superiority of our SS-Conv over alternative convolutions in terms of both accuracy and efficiency. We will make the implementation of SS-Conv publicly available.

2 Related Works

SE(3)-Equivariant Representation Learning SE(3)-equivariance is an important property in 3D computer vision. In earlier works, researchers ease the problem by focusing on SO(3)-equivariance, and design Spherical CNNs [6, 5] by stacking SO(3)-equivariant spherical convolutions which are implemented in the spherical harmonic domain. Recently, a series of works [19, 24, 7] build deep SE(3)-equivariant network based on steerable kernels, which are parameterized as linear combinations of basis kernels. Thomas *et al.* firstly propose Tensor Field Network (TFN) [19] to learn SE(3)-equivariant features on irregular point clouds, and later, Fuchs *et al.* present SE(3)-Transformer, which extends TFN with attention mechanism. However, those networks working on point clouds are required to compute kernels with respect to different input points inefficiently. To tackle this problem, steerable convolution [24] is proposed to work on regular volumetric data, so that basis kernels with respect to regular grids could be pre-computed; however, it still encounters challenging computational demands due to the ignorance of data sparsity. Compared to the above methods, our proposed Sparse Steerable Convolution aims at efficient SE(3)-equivariant representation learning for volumetric data, which is realized with sparse tensors to accelerate the computation.

Estimation and Tracking of Object Poses in 3D Space In the context of pose estimation, instance-level 6D pose estimation is a classical and well-developed task, for which a body of works are proposed. These works can be broadly categorized into three types: i) template matching [12] by constructing templates to search for the best matched poses; ii) 2D-3D correspondence methods [1, 14, 15, 18, 16], which establish 2D-3D correspondence via 2D keypoint detection [18, 16] or dense 3D coordinate predictions [1, 14, 15], followed by a PnP algorithm to obtain the target pose;

iii) direct pose regression [25, 13, 22] via deep networks. Recently, a more challenging task of category-level 6D pose and size estimation is formally introduced in [23], aiming at estimating poses of 3D unknown objects with respect to a categorical normalized object coordinate space (NOCS). The early work [23, 20] focuses on regression of NOCS maps, and the poses can be obtained by aligning them with the observed depth maps. Later, methods of direct pose regression are proposed based on fusion of pose-depentdent and pose-independent features [2], or on decoupled rotation mechanism [4]. Motivated by [23], Wang *et. al* propose the task of category-level 6D pose tracking, aiming for the small change of object poses between two adjacent frames in a sequence. In the paper, they also present 6-PACK, a tracker estimating the change of poses by matching keypoints of two frames.

3 Sparse Steerable Convolutional Networks

3.1 Background

3D Convolution A conventional 3D convolution can be formulated as follows:

$$f_{n+1}(\boldsymbol{x}) = [\kappa \star f_n](\boldsymbol{x}) = \int_{\mathbb{R}^3} \kappa(\boldsymbol{x}-\boldsymbol{y}) f_n(\boldsymbol{y}) d\boldsymbol{y}, \tag{1}$$

where $f_n(\boldsymbol{x}) \in \mathbb{R}^{K_n}$, $f_{n+1}(\boldsymbol{x}) \in \mathbb{R}^{K_{n+1}}$, and $\kappa : \mathbb{R}^3 \to \mathbb{R}^{K_{n+1} \times K_n}$ is a continuous learnable kernel.

SE(3)-Equivariance Given a transformation $\pi_n(\boldsymbol{g}) : \mathbb{R}^{K_n} \to \mathbb{R}^{K_n}$ for a 3D rigid motion $\boldsymbol{g} \in$SE(3), a 3D convolution in Eq. (1) is SE(3)-equivariant if there exists a transformation $\pi_{n+1}(\boldsymbol{g}) : \mathbb{R}^{K_{n+1}} \to \mathbb{R}^{K_{n+1}}$ such that

$$[\pi_{n+1}(\boldsymbol{g}) f_{n+1}](\boldsymbol{x}) = [\kappa \star [\pi_n(\boldsymbol{g}) f_n]](\boldsymbol{x}). \tag{2}$$

Such a SE(3)-equivariant convolution is the so-called *steerable convolution* (***ST-Conv***) [24], since the features can be steered by $\pi_{n+1}(\boldsymbol{g})$ in the output feature space.

In general, the transformation $\pi_n(\boldsymbol{g})$ is a group representation of SE(3), which satisfies $\pi_n(\boldsymbol{g_1 g_2}) = \pi_n(\boldsymbol{g_1})\pi_n(\boldsymbol{g_2})$. If $\boldsymbol{g}$ is decomposed into a 3D rotation $\boldsymbol{r} \in$ SO(3) and a 3D translation $\boldsymbol{t} \in \mathbb{R}^3$, written as $\boldsymbol{g} = \boldsymbol{tr}$, $\pi_n(\boldsymbol{g})$ can be defined as follows:

$$[\pi_n(\boldsymbol{g}) f_n](\boldsymbol{x}) = [\pi_n(\boldsymbol{tr}) f_n](\boldsymbol{x}) := \rho_n(\boldsymbol{r}) f_n(\boldsymbol{r}^{-1}(\boldsymbol{x}-\boldsymbol{t})), \tag{3}$$

where $\rho_n(\boldsymbol{r}) : \mathbb{R}^{K_n} \to \mathbb{R}^{K_n}$ is a SO(3) representation.

Rotation-Steering Constraint To guarantee SE(3)-equivariance in Eq. (2), it can be derived that the kernel κ of 3D convolution must be rotation-steerable [24], which satisfies the following constraint:

$$\kappa(\boldsymbol{rx}) = \rho_{n+1}(\boldsymbol{r}) \kappa(\boldsymbol{x}) \rho_n(\boldsymbol{r})^{-1}. \tag{4}$$

Irreducible Feature $\rho_n(\boldsymbol{r})$ is a SO(3) representation, which can be decomposed into F_n *irreducible representations* as follows:

$$\rho_n(\boldsymbol{r}) = \boldsymbol{Q}^T [\bigoplus_{i=0}^{F_n} D^{l_i}(\boldsymbol{r})] \boldsymbol{Q}, \tag{5}$$

where $\boldsymbol{Q}$ is a $K_n \times K_n$ change-of-basis matrix, $D^{l_i}(\boldsymbol{r})$ is the $(2l_i+1) \times (2l_i+1)$ irreducible Wigner-D matrix [8] of order l_i ($l_i = 0, 1, 2, ...$), and $\bigoplus$ represents block-diagonal construction of $\{D^{l_i}(\boldsymbol{r})\}$, so that $K_n = \sum_{i=0}^{F_n} 2l_i + 1$. Based on Eq. (3), $f_n(\boldsymbol{x})$ can be constructed by stacking F_n *irreducible features* $\{f_n^i(\boldsymbol{x}) \in \mathbb{R}^{2l_i+1}\}$; each $f_n^i(\boldsymbol{x})$ is associated with a $D^{l_i}(\boldsymbol{r})$. When $l_i = 0$, $D^0(\boldsymbol{r}) = 1$, so that $f_n^i(\boldsymbol{x})$ is a scalar, invariant to any rotation; when $l_i > 0$, $f_n^i(\boldsymbol{x})$ is a vector which can be rotated by $D^{l_i}(\boldsymbol{r})$.

3.2 Sparse Steerable Convolution

The ST-Conv introduced above enjoys the property of SE(3)-equivariance. However, as discussed in Sec. 1, it suffers from heavy computations as conventional 3D convolution does. Motivated by recent success of sparse convolution (***SP-Conv***) [9], we propose a novel design of *Sparse Steerable Convolution* (***SS-Conv***) with sparse tensors, which takes the natural sparsity of 3D data into account while strictly keeping the steerability of features.

Specifically, assuming κ is a discretized, $s \times s \times s$, cubic kernel with grid sites $\boldsymbol{S} = \{-\frac{s-1}{2}, ..., -1, 0, 1, ..., \frac{s-1}{2}\}^3$ (s is odd), our proposed SS-Conv can be formulated as follows:

$$f_{n+1}(\boldsymbol{x}) = [\kappa \star f_n](\boldsymbol{x}) = \begin{cases} \sum_{\boldsymbol{x}-\boldsymbol{y}\in\boldsymbol{S},\sigma_n(\boldsymbol{y})=1} \kappa(\boldsymbol{x}-\boldsymbol{y}) f_n(\boldsymbol{y}), & \text{if } \sigma_{n+1}(\boldsymbol{x}) = 1 \\ \boldsymbol{0}, & \text{if } \sigma_{n+1}(\boldsymbol{x}) = 0 \end{cases} \tag{6}$$
$$s.t. \quad \forall \boldsymbol{r} \in SO(3), \kappa(\boldsymbol{r}\boldsymbol{x}) = \rho_{n+1}(\boldsymbol{r})\kappa(\boldsymbol{x})\rho_n(\boldsymbol{r})^{-1},$$

where $\sigma_n(\boldsymbol{x})$ represents the state of site $\boldsymbol{x}$ in the feature space $\mathbb{R}^{K_n}$. $\sigma_n(\boldsymbol{x}) = 0$ denotes an inactive state at $\boldsymbol{x}$, where $f_n(\boldsymbol{x})$ is in its ground state; when $f_n(\boldsymbol{x})$ is beyond the ground state, this site would be activated as $\sigma_n(\boldsymbol{x}) = 1$. In SS-Conv, we set the ground state as a zero vector.

Compared with ST-Conv, our sparse version is accelerated in two ways: i) convolutions are conducted at activated output sites, not on the whole 3D volume, where the number of active sites only takes a small proportion; ii) in the receptive field of each activated output site, only input features at sites with active state are convolved. We introduce the detailed implementation of our SS-Conv as follows.

3.2.1 Rotation-Steering kernel Construction

The key to satisfy the rotation-steering constraint (4) is to control the angular directions of feature vectors, and recent research shows that the spherical harmonics $Y^J = \{Y_j^J\}_{j=-J}^J$ give the unique and complete solution [24]. Linear combination of the basis kernels based on spherical harmonics produces the rotation-steering convolutional kernel κ.

For simplicity, we firstly consider both input and output features as individual irreducible ones of orders l and k, respectively, and the kernel κ^{kl} is a linear combination of basis kernels $\kappa^{kl,Jm}$:

$$\kappa^{kl}(\boldsymbol{x}) = \sum_{J=|k-l|}^{k+l} \sum_m w^{kl,Jm} \kappa^{kl,Jm}(\boldsymbol{x}), \tag{7}$$

where

$$\kappa^{kl,Jm}(\boldsymbol{x}) = \sum_{j=-J}^{J} \varphi^m(\|\boldsymbol{x}\|) Y_j^J(\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}) \boldsymbol{Q}_j^{kl}. \tag{8}$$

$\boldsymbol{Q}_j^{kl}$ is a $(2k+1) \times (2l+1)$ change-of-basis matrix, also known as Clebsch-Gordan coefficients, and φ^m is a continuous Gaussian radial function: $\varphi^m(\|\boldsymbol{x}\|) = e^{-\frac{1}{2}(\|\boldsymbol{x}\|-m)^2/\epsilon^2}$. In the basis kernel $\kappa^{kl,Jm}$ (8), Y^J controls the angular direction, while φ^m controls the radial direction; then $\{\kappa^{kl,Jm}\}$ are linearly combined by learnable coefficients $\{w^{kl,Jm}\}$ as in Eq. (7), to further adjust the radial direction, the only degree of freedom in the process of optimization. Accordingly, the angular direction is totally controlled by Y^J, such that the kernel constraint is strictly followed. In addition, the total number of learnable parameters in Eq. (7) is $[2min(k,l)+1]m$, which is, in practice, marginally less than that of conventional 3D convolution, which has $(2k+1)(2l+1)$ parameters.

Finally, assuming that the input and output features are stacked irreducible features, whose orders are $\{l_1, \cdots, l_{F_n}\}$ and $\{k_1, \cdots, k_{F_{n+1}}\}$ respectively, the whole rotation-steering kernel can be constructed as follows:

$$\kappa(\boldsymbol{x}) = \begin{bmatrix} \kappa^{k_1 l_1}(\boldsymbol{x}) & \cdots & \kappa^{k_1 l_{F_n}}(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ \kappa^{k_{F_{n+1}} l_1}(\boldsymbol{x}) & \cdots & \kappa^{k_{F_{n+1}} l_{F_n}}(\boldsymbol{x}) \end{bmatrix}. \tag{9}$$

3.2.2 Site State Definition

The key enabling the efficiency of SS-Conv lies in the definition of active sites. In general, for a grid site $\boldsymbol{x}$, if any of input sites in its receptive field are active, this site will be activated, and convolution at this site will be conducted; otherwise, this site will keep inactive, meaning that its feature will be directly set as a zero vector (representing the ground state) without convolutional operation. We formulate the above definition of site state at $\boldsymbol{x}$ as follows:

$$\sigma_{n+1}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \sum_{\boldsymbol{x}-\boldsymbol{y}\in\boldsymbol{S}} \sigma_n(\boldsymbol{y}) > 0 \\ 0. & \text{if } \sum_{\boldsymbol{x}-\boldsymbol{y}\in\boldsymbol{S}} \sigma_n(\boldsymbol{y}) = 0 \end{cases} \tag{10}$$

The number of active sites defined in this way will increase layer-by-layer, enabling long-range message transfer. However, when stacking dozens or even hundreds of convolutions, the rapid growth rate of active sites would result in heavy computational burden and the so-called *"submanifold dilation problem"* [9]. To alleviate this problem, we follow [9] and consider another choice of state definition in our SS-Conv, which keeps the output state consistent with the input one at a same grid site, *i.e.*, $\sigma_{n+1}(\boldsymbol{x}) = \sigma_n(\boldsymbol{x})$. This kind of SS-Conv without dilation makes it possible to construct a deep but efficient network for sparse volumetric data, and we term it as "*Submanifold SS-Conv*". In practice, we mix general SS-Convs and Submanifold SS-Convs in an alternating manner to achieve high performance and efficiency.

3.2.3 Sparse Convolutional Operation

SS-Conv is achieved with sparse tensors. A sparse tensor can be represented as $(\boldsymbol{H}_{n+1}, \boldsymbol{F}_{n+1})$, where $\boldsymbol{H}_{n+1}$ is a hash table recording the coordinates of those sites with active state, and $\boldsymbol{F}_{n+1}$ is a feature matrix. $\boldsymbol{H}_{n+1}$ and $\boldsymbol{F}_{n+1}$ correspond to each other row-by-row; that is, if $r_{n+1,\boldsymbol{x}}$ is the row number of $\boldsymbol{x}$ in $\boldsymbol{H}_{n+1}$, then $\boldsymbol{F}_{n+1}[r_{n+1,\boldsymbol{x}}] = f_{n+1}(\boldsymbol{x})$.

After obtaining $\{\sigma_{n+1}(\boldsymbol{x})\}$, the output hash table $\boldsymbol{H}_{n+1}$ can be generated; the next target is to compute the values of $\boldsymbol{F}_{n+1}$. Specifically, we firstly initialize $\boldsymbol{F}_{n+1}$ to zeros; then the feature vectors in $\boldsymbol{F}_{n+1}$ are updated via the following algorithm:

ALGORITHM 1: Sparse Steerable Convolution

Input: $(\boldsymbol{H}_n, \boldsymbol{F}_n), (\boldsymbol{H}_{n+1}, \boldsymbol{F}_{n+1}), \{\kappa(\boldsymbol{s}) : \boldsymbol{s} \in \boldsymbol{S}\}$
Output: $(\boldsymbol{H}_{n+1}, \boldsymbol{F}_{n+1})$

```
R = {R_s = ∅ : s ∈ S}                          // Initialize the rule book R.
for x in H_{n+1} do                            // Construct the rule book R.
    for y in H_n do
        if s = x − y ∈ S:
            Append (r_{n+1,x}, r_{n,y}) to R_s.   // r_{n+1,x} is the row number of x in H_{n+1}.
    end for                                    // r_{n,y} is the row number of y in H_n.
end for
for R_s in R do                                // Update F_{n+1}.
    for (r_{n+1,x}, r_{n,y}) in R_s do
        F_{n+1}[r_{n+1,x}] = F_{n+1}[r_{n+1,x}] + κ(s) × F_n[r_{n,y}]
    end for
end for
```

This process can be divided into two steps. The first step is to construct a rule book $\boldsymbol{R} = \{\boldsymbol{R}_{\boldsymbol{s}} : \boldsymbol{s} \in \boldsymbol{S}\}$, where an active output site $\boldsymbol{x}$ is paired with an active input $\boldsymbol{y}$ in each $\boldsymbol{R}_{\boldsymbol{s}}$, if $\boldsymbol{x} - \boldsymbol{y} = \boldsymbol{s}$. The second step is to update $\boldsymbol{F}_{n+1}$ according to the paired relationships recorded in $\boldsymbol{R}$; for example, if the paired relationship of output $\boldsymbol{x}$ and input $\boldsymbol{y}$ is recorded in $\boldsymbol{R}_{\boldsymbol{s}}$, the current $f_{n+1}(\boldsymbol{x})$ will be updated by adding the multiplication of $f_n(\boldsymbol{y})$ and $\kappa(\boldsymbol{s})$. In this process, the construction of $\boldsymbol{R}$ is very critical, which helps to implement the second step by matrix-matrix multiply-add operations on GPUs efficiently.

3.3 Normalization and Activation

As conventional CNNs do, SS-Convs are also followed by normalization and activation, *i.e.*, *Activation*(*Norm*($[\kappa \star f_n](\boldsymbol{x})$)). Those operations of normalization and activation are required to be specially designed, not to break the SE(3)-equivariance of features. Since each SE(3)-equivariant feature is formed by stacking irreducible ones, without loss of generality, we take as an example an irreducible feature $\boldsymbol{f}(\boldsymbol{x})$ with order l, so that the normalization can be formulated as follows:

$$\mathit{Norm}(f(\boldsymbol{x})) = \begin{cases} (f(\boldsymbol{x}) - \mu)/(\sqrt{\frac{1}{N}\sum_{\sigma(\boldsymbol{x})=1} \|f(\boldsymbol{x}) - \mu\|^2 + \epsilon}), & l = 0 \\ f(\boldsymbol{x})/(\sqrt{\frac{1}{N}\sum_{\sigma(\boldsymbol{x})=1} \|f(\boldsymbol{x})\|^2 + \epsilon}), & l > 0 \end{cases} \tag{11}$$

where $\mu = \frac{1}{N}\sum_{\sigma(\boldsymbol{x})=1} f(\boldsymbol{x})$, N is the number of active sites, and ϵ is a very small constant. For the activation of $f(\boldsymbol{x})$, if $l = 0$, *ReLU* can be chosen to increase non-linearity; if $l > 0$, we follow [24] and multiply to $f(\boldsymbol{x})$ a scalar, which is learned by a SS-Conv and applied to the *Sigmoid* function:

$$\mathit{Activation}(f(\boldsymbol{x})) = \begin{cases} \mathit{ReLU}(f(\boldsymbol{x})), & l = 0 \\ \mathit{Sigmoid}([\kappa^{0l} \star f](\boldsymbol{x}))f(\boldsymbol{x}). & l > 0 \end{cases} \tag{12}$$

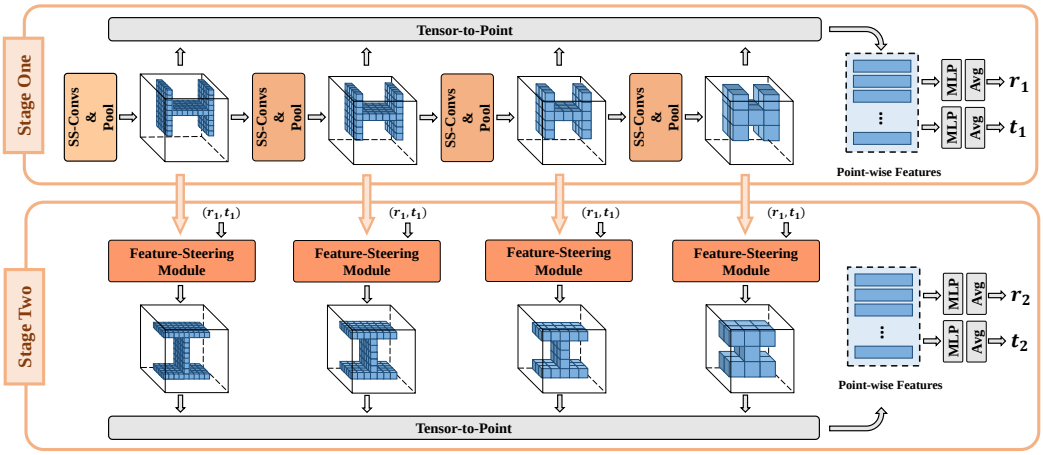


Figure 2: An illustration of network architecture for instance-level 6D object pose estimation.

The above normalization and activation operations are both SE(3)-equivariant, since a feature vector multiplying any scalar keeps its equivariance; when applying them to features formed by numerous irreducible ones, we treat each irreducible member individually to ensure the equivariance.

4 Applications for Estimation and Tracking of Object Poses in 3D Space

4.1 Instance-level 6D Object Pose Estimation

Given an RGB-D image of a cluttered scene, instance-level 6D pose estimation is to estimate the 6D poses of known 3D objects with respect to the camera coordinate system. As introduced in Sec. 3.1, a 6D pose $\boldsymbol{g} \in \mathrm{SE}(3)$ can be decomposed into a 3D rotation $\boldsymbol{r} \in \mathrm{SO}(3)$ and a 3D translation $\boldsymbol{t} \in \mathbb{R}^3$, which makes SS-Conv based network well suited for this task, due to: i) SS-Convs provide strong SE(3)-equivariant features to decode a precise 6D pose; ii) the steerability of feature maps helps to enable a second stage of pose refinement. Therefore, we propose an efficient SS-Conv based general pipeline for 6D pose estimation, as depicted in Fig. 2.

Specifically, we firstly segment out the objects of interest by an off-the-shelf model of instance segmentation, assigning each object with an RGB segment and a cropped point cloud; then each 3D object is voxelized and represented by a sparse tensor $(\boldsymbol{H}_0, \boldsymbol{F}_0)$, where each feature in $\boldsymbol{F}_0$ is a $4-$dimensional vector, containing RGB values and a constant "1". For the input tensor, we set the site active, if the quantified grid centered at this site encloses any points, and average point features of those enclosed by a same grid. $(\boldsymbol{H}_0, \boldsymbol{F}_0)$ is then fed into our pose estimation network, where the estimation could be achieved in the following two stages.

In the first stage, we construct an efficient SS-Conv based backbone, which extracts hierarchical SE(3)-equivariant feature maps, represented in the form of sparse tensors $\{(\boldsymbol{H}_n, \boldsymbol{F}_n)\}$. Those feature tensors are used for interpolation of multi-level point-wise features by using Tensor-to-Point Modules, proposed in [10], transforming features of discretized grid sites to those of real-world point coordinates. Each point feature is fed into two separate MLPs, regressing a point offset and a rotation, respectively, where the addition of the point coordinate and its offset generates a translation. The initially predicted pose $(\boldsymbol{r}_1, \boldsymbol{t}_1)$ of this stage is obtained by averaging point-wise predictions.

In the second stage, we refine the pose $(\boldsymbol{r}_1, \boldsymbol{t}_1)$ by learning a residual pose $(\boldsymbol{r}_2, \boldsymbol{t}_2)$, wherein a Feature-Steering module is designed, generating transformed features $\{(\boldsymbol{H}'_n, \boldsymbol{F}'_n)\}$ by efficiently steering hierarchical backbone features $\{(\boldsymbol{H}_n, \boldsymbol{F}_n)\}$ individually with $(\boldsymbol{r}_1, \boldsymbol{t}_1)$. Again we interpolate point-wise features from $\{(\boldsymbol{H}'_n, \boldsymbol{F}'_n)\}$, and average point-wise predictions to obtain $(\boldsymbol{r}_2, \boldsymbol{t}_2)$. Finally, the predicted 6D pose is then updated as $(\boldsymbol{r}_1\boldsymbol{r}_2, \boldsymbol{t}_1 + \boldsymbol{r}_1\boldsymbol{t}_2)$. Additionally, owing to the novel SE(3)-Steering modules, this stage can be iteratively repeated, generating finer and finer poses.

4.1.1 The Feature-Steering Module

Feature-Steering module in the pipeline is to transform $(\boldsymbol{H}_n, \boldsymbol{F}_n)$ of the backbone to $(\boldsymbol{H}'_n, \boldsymbol{F}'_n)$, where a rigid transformation of $\boldsymbol{H}_n$ with $(\boldsymbol{r}, \boldsymbol{t})$ and a rotation of $\boldsymbol{F}_n$ with $\rho(\boldsymbol{r})$ are included. Specifically, for $\boldsymbol{F}_n$, we compute $\rho(\boldsymbol{r})$ as defined in (5) and rotate $\boldsymbol{F}_n$ by matrix multiplication; for $\boldsymbol{H}_n$, we convert the sites in it to the real-world point coordinates, which are then applied to a rigid transformation of $(\boldsymbol{r}, \boldsymbol{t})$ and re-voxelized as grid sites. The same new sites are merged to a unique one, while their features are averaged. We also use two another SS-Convs, each followed by steerable normalization and activation, to enrich the new features and generate the final steered $(\boldsymbol{H}'_n, \boldsymbol{F}'_n)$.

4.2 Category-level 6D Object Pose and Size Estimation

Category-level 6D pose and size estimation is formally introduced in [23]. This is a more challenging task, which aims to estimate categorical 6D poses of unknown objects, and also the 3D object sizes. To tackle this problem, we use a similar network as that in Fig. 2, and make some adaptive modifications: i) for each stage in Fig. 2, we add another two separate MLPs for point-wise predictions of 3D sizes and point coordinates in the canonical space, respectively; ii) in each Feature-Steering module, the real-world coordinates of all 3D objects are also scaled by their predicted 3D sizes to be enclosed within a unit cube, for estimating more precise poses.

4.3 Category-level 6D Object Pose Tracking

Motivated by the above task of categorical pose estimation, category-level 6D pose tracking is also proposed to estimate the small change of 6D poses in two adjacent RGB-D frames of an image sequence [21]. Due to the available pose of the previous frame, the target object can be roughly located in the current frame, avoiding the procedures of object detection or instance segmentation in images. However, without a precise mask, the estimation of small pose change from noisy 3D data is a big challenge for deep networks. Our SS-Conv based network also surprisingly performs well in such noisy data, even though we only conduct one-stage pose estimation that achieves real-time tracking. For more details, one may refer to the supplementary material.

5 Experiments

Datasets We conduct experiments on the benchmark LineMOD dataset [11] for instance-level 6D pose estimation, which consists of 13 different objects. For both category-level 6D pose estimation and tracking, we experiment on REAL275 dataset [23], which is a more challenging real-world dataset with $4,300$ training images and $2,750$ testing ones, containing object instances of 6 categories. Following [23, 21], we augment the training data of REAL275 with synthetic RGB-D images.

Evaluation Metrics For instance-level task, we follow [22] and evaluate the results of LineMOD dataset on ADD(S) metric. For the category-level task, we report the mean Average Precision (mAP) of intersection over union (IoU) and $n^\circ m$ cm, following [23]; mean rotation error ($\boldsymbol{r}_{err}$) in degrees and mean translation error ($\boldsymbol{t}_{err}$) in centimeters are also reported for pose tracking. Additionally, we compare the numbers of parameters (#Param) and the running speeds (FPS) for different models. Testing is conducted on a server with GeForce RTX 2080ti GPU for a batchsize of 32, and FPS is computed by averaging the time cost of forward propagation on the whole dataset.

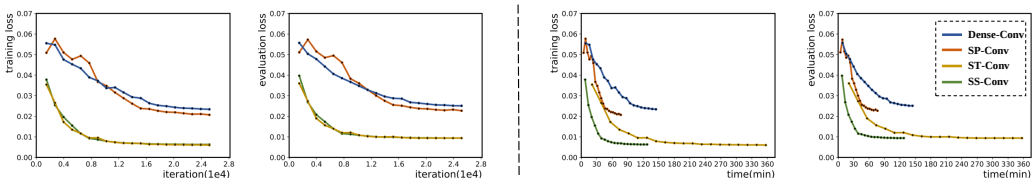
5.1 Comparisons with Different 3D Convolutions

We firstly conduct experiments to compare our proposed SS-Conv with other kinds of 3D convolutions, including conventional 3D convolution (Dense-Conv), sparse convolution (SP-Conv) [9], and steerable convolution (ST-Conv) [24], on the LineMOD dataset for instance-level 6D pose estimation. Among those convolutions, SP-Conv improves the speed of Dense-Conv by considering data sparsity and turns out to be efficient in 3D object detection, while ST-Conv constructs rotation-steering kernels and realizes the convolutional operation based on Dense-Conv.

To meet various computational demands of different convolutions, those experiments are conducted on a light plain architecture in the same experimental settings, for a fair comparison. The architecture consists of 12 convolutional layers, of which the kernel sizes are all set as $3 \times 3 \times 3$; feature channels are kept consistent when applying different convolutions. We use ADAM to train the networks for a

Table 1: Quantitative comparisons of different 3D convolutions on the LineMOD dataset [11].

Conv	ADD(S) ↑	FPS ↑	#Param ↓
Dense-Conv	46.5	224	26.2 M
SP-Conv	62.8	486	26.2 M
ST-Conv	92.8	148	3.6 M
SS-Conv	93.5	293	3.6 M



(a) **(b)**

Figure 3: Training of plain networks with different convolutions on the LineMOD dataset [11].

total of $30,000$ iterations, with an initial learning rate of 0.01; learning rate is halved every $1,500$ iterations. We voxelize the input segmented objects into $64 \times 64 \times 64$ dense/sparse grids, and set the training batchsize as 16.

Quantitative results of different convolutions are listed in Table 1, which confirms the advantages of our SS-Conv in both performance and efficiency. In terms of performance, SS-Conv achieves comparable results on LineMOD dataset as ST-Conv does, which marginally outperforms those of Dense-Conv and SP-Conv, indicating the importance of SE(3)-equivariant feature learning on pose estimation. Although the convolutional kernel of SS-Conv is restricted under the rotation-steering constraint, its capacity of feature extraction is not weakened, and instead, strengthened by constantly preserving the relative poses of features layer-by-layer; as a result, the property of SE(3)-equivariance makes it natural for learned features to extract more information of object poses. Additionally, due to the kernel constraint, the number of learnable parameters of SS-Conv/ST-Conv is much smaller than those of the other two convolutions, which is also listed in Table 1. In terms of efficiency, SS-Conv based network doubles the FPS of ST-Conv based one, and becomes more efficient and flexible to be applied to complex systems. The efficiency of SS-Conv can also be verified in Fig. 3, where we visualize the behaviors of the four convolutions in the process of training; as shown in the figure, the training time of SS-Conv based network is marginally shortened, compared to that of ST-Conv based one, although their loss curves both decrease consistently over iterations.

5.2 Comparisons with Existing Methods

Instance-level 6D Object Pose Estimation For the instance-level task, we compare the results of our SS-Conv based pipeline with existing methods on LineMOD dataset [11]. Quantitative results are shown in Table 2, where our two-stage pipeline outperforms all the existing methods and achieves a new state-of-the-art result of 99.2% on mean ADD(S) metric. We can also observe that the second stage of pose refinement with Feature-Steering modules in our pipeline indeed improves the predictions in the first stage, benefitting from the steerability of the feature spaces in SS-Convs.

Category-level 6D Object Pose and Size Estimation We conduct experiments on REAL275 [23] for the more challenging category-level task. Quantitative results in Table 3 confirm the advantage of our pipeline in the high-precision regime, especially on the precise metric of $5°5$cm, where we improve the state-of-the-art result in [4] from 28.5% to 41.7%. The second stage of pose refinement also plays an important role in this task, achieving remarkable improvements over the first stage.

Category-level 6D Object Pose Tracking We compare the results of our one-stage tracking pipeline with the baseline of 6-PACK [21] on REAL275 [23]. In 6-PACK, the relative pose between two frames is computed based on predicted keypoint pairs inefficiently, while our pipeline regresses the pose in a direct way. The results in Table 4 show that our pipeline outperforms 6-PACK on all the evaluation metrics, demonstrating the ability of SS-Conv based network for fine-grained pose estimation in noisy input data.

More implementation details and qualitative results are shown in the supplementary material.

Table 2: Quantitative comparisons of different methods on the LineMOD dataset [11] for instance-level 6D object pose estimation. The evaluation metric is ADD(S).

	Implicit[17] +ICP	SSD6D[13] +ICP	PointFusion [26]	DenseFusion [22]	DenseFusion (Iterative)[22]	G2L[3]	Ours w/o second stage	Ours
ape	20.6	65	70.4	79.5	92.3	96.8	92.9	**97.4**
bench.	64.3	80	80.7	84.2	93.2	96.1	97.4	**99.3**
camera	63.2	78	60.8	76.5	94.4	98.2	97.7	**99.5**
can	76.1	86	61.1	86.6	93.1	98.0	96.1	**99.6**
cat	72.0	70	79.1	88.8	96.5	99.2	98.8	**99.8**
driller	41.6	73	47.3	77.7	87.0	**99.8**	98.7	99.6
duck	32.4	66	63.0	76.3	92.3	97.7	91.1	**97.8**
egg.	98.6	100	99.9	99.9	99.8	**100.0**	**100.0**	99.9
glue	96.4	100	99.3	99.4	**100.0**	**100.0**	98.6	99.6
hole.	49.9	49	71.8	79.0	92.1	99.0	96.3	**99.4**
iron	63.1	78	83.2	92.1	97.0	**99.3**	98.7	99.2
lamp	91.7	73	62.3	92.3	95.3	99.5	99.5	**99.7**
phone	71.0	79	78.8	88.0	92.8	**98.9**	97.5	98.2
MEAN	64.7	79	73.7	86.2	94.3	98.7	97.2	**99.2**

Table 3: Quantitative comparisons of different methods on REAL275 dataset [23] for category-level 6D object pose and size estimation.

Method	mAP					
	IoU_{50}	IoU_{75}	5°2cm	5°5cm	10°2cm	10°5cm
NOCS [23]	78.0	30.1	7.2	10.0	13.8	25.2
SPD [20]	77.3	53.2	19.3	21.4	43.2	54.1
CASS [2]	77.7	–	–	23.5	–	58.0
FS-Net [4]	**92.2**	63.5	–	28.2	–	60.8
Ours w/o second stage	79.5	58.7	19.2	25.2	35.1	49.9
Ours	79.1	**63.7**	**31.5**	**41.7**	**45.8**	**61.7**

6 Discussions

The studied problems of object pose estimation and tracking in 3D space are very important to many real-world applications, including augmented reality, robotic grasping, and autonomous driving. By precisely predicting object poses in the 3D space, virtual contents could be seamlessly embedded in real environments, creating fascinating personal experience; on the contrary, less precise predictions may cause property loss and even life threat, especially in autonomous driving. The contributed solution based on SS-Conv would improve the overall level of safety.

In the paper, while we only demonstrate the advantages of SS-Conv on an object-centric pipeline, its capacity for processing 3D scenes with small regions is also revealed in tracking. In future work, we will work on the investigation of its potentials for tasks on large-scale scenes.

Table 4: Quantitative comparisons of different methods on REAL275 dataset [23] for category-level 6D object pose tracking.

Method	Metric	bottle	bow	camera	can	laptop	mug	MEAN
6PACK [21]	$5°5cm \uparrow$	24.5	55.0	10.1	22.6	63.5	24.1	33.3
	$IoU_{25} \uparrow$	91.1	100.0	87.6	92.6	98.1	95.2	94.1
	$R_{err} \downarrow$	15.6	5.2	35.7	13.9	4.7	21.3	16.1
	$T_{err} \downarrow$	4.0	1.7	5.6	4.8	2.5	2.3	3.5
Ours	$5°5cm \uparrow$	70.3	60.6	10.6	49.9	87.7	47.9	54.5
	$IoU_{25} \uparrow$	93.5	99.9	99.9	99.8	99.8	99.9	98.8
	$R_{err} \downarrow$	3.7	4.6	9.8	4.6	3.0	5.6	5.2
	$T_{err} \downarrow$	1.9	1.2	2.0	2.7	2.4	1.1	1.9

References

[1] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014.

[2] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. Learning canonical shape space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11973–11982, 2020.

[3] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, and Ales Leonardis. G2l-net: global to local network for real-time 6d pose estimation with embedding vector features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4233–4242, 2020.

[4] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, Linlin Shen, and Ales Leonardis. Fs-net: Fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism. *arXiv preprint arXiv:2103.07054*, 2021.

[5] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *International Conference on Learning Representations*, 2018.

[6] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

[7] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[8] Robert Gilmore. *Lie groups, physics, and geometry: an introduction for physicists, engineers and chemists*. Cambridge University Press, 2008.

[9] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[10] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020.

[11] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *2011 international conference on computer vision*, pages 858–865. IEEE, 2011.

[12] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[13] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE international conference on computer vision*, pages 1521–1529, 2017.

[14] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In *Proceedings of the IEEE international conference on computer vision*, pages 954–962, 2015.

[15] Frank Michel, Alexander Kirillov, Eric Brachmann, Alexander Krull, Stefan Gumhold, Bogdan Savchynskyy, and Carsten Rother. Global hypothesis generation for 6d object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 462–471, 2017.

[16] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.

[17] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018.

[18] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018.

[19] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[20] Meng Tian, Marcelo H Ang, and Gim Hee Lee. Shape prior deformation for categorical 6d object pose and size estimation. In *European Conference on Computer Vision*, pages 530–546. Springer, 2020.

[21] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066. IEEE, 2020.

[22] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019.

[23] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[24] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018.

[25] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[26] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sec. 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sec. 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]