Learning Viewpoint-Agnostic Visual Representations by Recovering Tokens in 3D Space

Anonymous Author(s)
Affiliation
Address
email

Abstract

Humans are remarkably flexible in understanding viewpoint changes due to visual cortex supporting the perception of 3D structure. In contrast, most of the computer vision models that learn visual representation from a pool of 2D images often fail to generalize over novel camera viewpoints. Recently, the vision architectures have shifted towards convolution-free architectures, visual Transformers, which operate on tokens derived from image patches. However, these Transformers do not perform explicit operations to learn viewpoint-agnostic representation for visual understanding, as in convolutions. To this end, we propose a 3D Token Representation Layer (3DTRL) that estimates the 3D positional information of the visual tokens and leverages it for learning viewpoint-agnostic representations. The key elements of 3DTRL include a pseudo-depth estimator and a learned camera matrix to impose geometric transformations on the tokens. These enable 3DTRL to recover the 3D positional information of the tokens from 2D patches. In practice, 3DTRL is easily plugged-in into a Transformer. Our experiments demonstrate the effectiveness of 3DTRL in many vision tasks including image classification, multi-view video alignment, and action recognition. The models with 3DTRL outperform their backbone Transformers in all the tasks with minimal added computation.

1 Introduction

Over the past few years, computer vision models have developed rapidly from CNNs [4, 23, 51] to now Transformers [14, 21, 52]. With these models, we can now accurately classify objects in an image, align image frames among video pairs, classify actions in videos, and more. Despite their success, many of the models neglect that the world is 3D and do not extend beyond the XY image plane [18]. While humans can readily estimate the 3D structure of a scene from 2D pixels of an image, most of the existing vision models with 2D images do not take the 3D structure of the world into consideration. This is one of the reason why humans are able to recognize objects in images and actions in videos regardless of their viewpoint, but the vision models often fail to generalize over novel viewpoints [9, 18, 37].

Consequently, in this paper, we develop an approach to learn viewpoint-agnostic representations for a robust understanding of the visual data. Naive solutions to obtain viewpoint-agnostic representation would be either supervising the model with densely annotated 3D data, or learning representation from a large scale 2D datasets with samples encompassing different viewpoints. Given the fact that such high quality data are expensive to acquire and hard to scale, an approach with a higher sample efficiency without 3D supervision is desired.

To this end, we propose a 3D Token Representation Layer (3DTRL), incorporating 3D camera transformations into the recent successful visual Transformers [5, 14, 32, 52]. 3DTRL first recovers camera-centered 3D coordinates of each token by depth estimation. Then 3DTRL estimates a camera

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

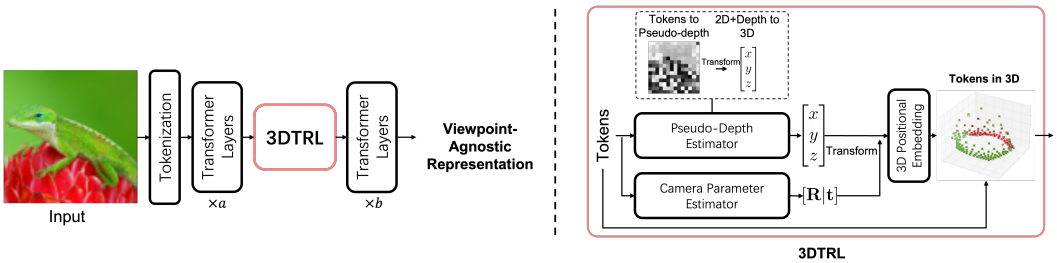


Figure 1: Overview of proposed 3DTRL. **Left**: 3DTRL is a module inserted in between Transformer layers. **Right**: 3DTRL has three parts, Pseudo-Depth Estimator, Camera Parameter Estimator, and 3D Positional Embedding layer. Within Pseudo-Depth Estimator, we first estimate depth of each token and then calculate 3D coordinates from 2D locations and depth.

matrix to transform these camera-centered coordinates to a 3D world space. In this world space, 3D locations of the tokens are absolute and view-invariant, which contain important information for learning viewpoint-agnostic representations. Therefore, 3DTRL incorporates such 3D positional information in the Transformer backbone in the form of 3D positional embeddings, and generates output tokens with 3D information. Unlike visual Transformers only relying on 2D positional embedding, models with 3DTRL are more compliant with learning viewpoint-agnostic representations.

We conduct extensive experiments on various vision tasks to confirm the effectiveness of 3DTRL. Our 3DTRL outperforms the Transformer backbones on 3 image classification, 5 multi-view video alignment, and 2 multi-view action recognition datasets in their respective tasks. Moreover, 3DTRL is a light-weighted, plug-and-play module that achieves the above improvements with minimal (2% computation and 4% parameters) overhead.

In summary, we present a learnable, differentiable layer 3DTRL that efficiently and effectively learns viewpoint-agnostic representations.

2 Background: Pinhole Camera Model

3DTRL is based on the standard pinhole camera model widely used in computer vision. Thus, we first briefly review the pinhole camera model. In homogeneous coordinate system, given a point with world coordinate p^{world}, a camera projects a pixel at p on an image by:

$$p = \mathbf{K}\ [\ \mathbf{R}|\mathbf{t}\ ]\ p^{\text{world}}, \tag{1}$$

where $\mathbf{K}$ is the intrinsic matrix and $[\ \mathbf{R}|\mathbf{t}\ ]$ is the extrinsic matrix. $\mathbf{K}$ is further represented by

$$\mathbf{K} = \begin{bmatrix} c & 0 & u \\ 0 & c & v \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where c is the focal length and (u, v) are the pixel coordinates. In this work, we explore visual understanding in multi-view setting, thus aiming at learning viewpoint-agnostic representations. In this setting, a scene may be captured with different cameras positioned at non-identical locations and viewing angles. Here, the world coordinate p^{world} is the same across all the cameras while pixel projection p is different across cameras. We focus on how to estimate the world coordinates p^{world} from their corresponding image pixels at p. Estimating p^{world} from p involves two transformations that correspond to the *inverse* of $\mathbf{K}$ and $[\ \mathbf{R}|\mathbf{t}\ ]$ which might not be known beforehand. Thus, we *estimate* them from image patches instead, which is a key procedure in 3DTRL.

3 3D Token Representation Layer (3DTRL)

In this section, we detail how 3DTRL estimates 3D positional information in a Transformer and is integrated. We will introduce 3DTRL for image analysis, then we adapt it to video models.

3.1 Overview

3DTRL is a simple yet effective plug-in module that can be inserted in between the layers of visual Transformers (Figure 1 left). Given a set of input tokens S, 3DTRL returns a set of tokens with

3D information. The number of tokens and their dimensionality are kept unchanged. Within the module (Figure 1 right), 3DTRL first performs 3D estimation using two estimators: (1) pseudo-depth estimator and (2) camera parameter estimator. Then the tokens are associated with their recovered world 3D coordinates, which are transformed from estimated depth and camera matrix. Finally, 3DTRL generate 3D positional embeddings from these world coordinates and combine them with input S to generate the output of 3DTRL.

In the aspect that we insert 3DTRL in between the Transformer model, 3DTRL implicitly leverages the Transformer layers *before* it to be a part of the 3D estimators, and layers *after* that to be the actual 3D feature encoder (Figure 1 left). This avoids adding a large number of parameters while resulting in reasonable estimations. We empirically find that placing 3DTRL at a shallow-medium layer of the network yields better results (See Section 4.3), which is a trade-off between model capacity for estimation and 3D feature encoding.

3.2 3D Estimation of the Input Tokens

3DTRL first estimates both the camera-centered 3D coordinates of each token using depth estimation and a camera matrix shared by all the tokens of an image. Then, the camera-centered 3D coordinates are transformed to the "world" coordinates by the camera matrix.

Pseudo-depth Estimation. Given input tokens $S = \{s_1, \ldots, s_N\} \in \mathbb{R}^{N \times m}$, 3DTRL first performs pseudo-depth estimation of each token s_n. The pseudo-depth estimator is a function $f : \mathbb{R}^m \to \mathbb{R}$ that outputs the depth $d_n = f(s_n)$ of each token individually. We implement f using a 2-layer MLP. We call this pseudo-depth estimation since it is similar to depth estimation from monocular images but operates at a very coarse scale, given that each token corresponds to an image patch rather than a single pixel in Transformer.

After pseudo-depth estimation, 3DTRL transforms the pseudo-depth map to camera-centered 3D coordinates. Recall that in ViT [14], an image X is decomposed into N patches $\{X_1, \ldots, X_N\} \in \mathbb{R}^{N \times P \times P \times 3}$, where $P \times P$ is the size of each image patch and the tokens S are obtained from a linear projection of these image patches. Thus, each token is initially associated with a 2D location on the image plane, denoted as (u, v). By depth estimation, 3DTRL associates each token with one more value d. Based on the pinhole camera model explained in Section 2, 3DTRL transforms u, v, d to a camera-centered 3D coordinate (x, y, z) by:

$$p_n^{\text{cam}} = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} u_n z_n \\ v_n z_n \\ z_n \end{bmatrix}, \text{ where } z_n = \frac{d_n}{\sqrt{u_n^2 + v_n^2 + c}}. \tag{3}$$

Since we purely perform the aforementioned estimation from monocular images and the the camera intrinsic matrix is unknown, we simply set c to a constant hyperparameter. To define coordinate system of 2D image plane (u, v), we set the center of the original image is the origin $(0, 0)$ for convenience, so that the image plane and camera coordinate system shares the same origin. We use the center of the image patch to represent its associate (u, v) coordinates.

We believe that this depth-based 3D coordinate estimation best leverages the known 2D geometry, which is beneficial for later representation learning. We later confirm this in our ablation study (in Section 4.3), in which we compare it against a variant directly estimating (x, y, z).

Camera Parameter Estimation. The camera parameters are required to transform the estimated camera-centered 3D coordinates p^{cam} to the world coordinate system. These camera parameters are estimated jointly from all input tokens S. This involves estimation of two matrices, a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 and translation matrix $\mathbf{t}$ through an estimator g. We implement g using a MLP. Specifically, we use a shared MLP stem to aggregate all the tokens into an intermediate representation. Then, we use two separated fully connected heads to estimate the parameters in $\mathbf{R}$ and $\mathbf{t}$ respectively. We note the camera parameter estimator as a whole: $[\, \mathbf{R} | \mathbf{t} \,] = g(S)$. To ensure $\mathbf{R}$ is mathematically valid, we first estimate the three values corresponding to raw, pitch and roll angles of the camera pose, and then convert them into a 3×3 rotation matrix.

Transform to World Coordinates. Now, with the estimated camera parameters, 3DTRL transforms estimated camera-centered coordinates p^{cam} into the world space, a 3D space where 3D coordinates of the tokens are absolute and viewpoint-invariant. Following the pinhole camera model, we recover p^{world} by

$$p_n^{\text{world}} = [\, \mathbf{R}^T | \mathbf{R}^T \mathbf{t} \,]\, p_n^{\text{cam}}. \tag{4}$$

3.3 Incorporating 3D Positional Information in Transformers

The last step of 3DTRL is to leverage the estimated 3D positional information in Transformer backbone. For this, we choose to adopt a typical technique of incorporating positional embedding that is already used in Transformers [14, 53]. In contrast to 2D positional embedding in ViTs [14], 3DTRL learns a 3D embedding function $h : \mathbb{R}^3 \rightarrow \mathbb{R}^m$ to transform estimated world coordinates p^{world} to positional embeddings p^{3D}. This 3D embedding function h is implemented using a two-layer MLP. Then, the obtained 3D positional embedding is incorporated in the Transformer backbone by combining it with the token representations. The outcome is the final token representations $\{s^{\text{3D}}\}$:

$$s_n^{\text{3D}} = s_n + p_n^{\text{3D}}, \text{ where } p^{\text{3D}} = h(p^{\text{world}}). \tag{5}$$

After 3D embedding, the resultant token representations are associated with a 3D space, thus enabling the remaining Transformer layers to encode viewpoint-agnostic token representations. We ablate other ways of incorporating the 3D positional information of the tokens in Section 4.3.

3.4 3DTRL in Video Models

Notably, 3DTRL can be also easily generalized to video models. For video models, the input to 3DTRL is a set of spatial-temporal tokens $\{S_1, \ldots, S_T\}$ corresponding to a video clip containing T frames, where $S_t = \{s_{t1}, \ldots, s_{tN}\}$ are N tokens from t-th frame. We simply extend our module to operate on an additional time dimension, where depth estimation and 3D positional embedding are done for each spatial-temporal tokens s_{tn} individually: $d_{tn} = f(s_{tn})$, $p_{tn}^{\text{3D}} = h(p_{tn}^{\text{world}})$. Camera parameters are estimated per input frame (S_t) in a dissociated manner, namely $[\ \mathbf{R}|\mathbf{t}\ ]_t = g(S_t)$, resulting in a total of T camera matrices per video. We investigate another strategy of camera estimation in the supplementary, where only one camera matrix is learned for all frames. However, our studies have substantiated the effectiveness of learning dissociated camera matrices per frame.

4 Experiments

We conduct extensive experiments to demonstrate the efficacy of viewpoint-agnostic representations learned by 3DTRL in multiple vision tasks: (i) image classification, (ii) multi-view video alignment, and (iii) video action classification.

4.1 Image Classification

In order to validate the power of 3DTRL, we first evaluate ViT [14] with 3DTRL for image classification task using CIFAR-10 [30], CIFAR-100 [30] and ImageNet-1K [13] datasets.

Training We use the training recipe of DeiT [52] for training our baseline Vision Transformer model on CIFAR and ImageNet datasets from *scratch*. We performed ablations to find an optimal location in ViTs where 3DTRL should be plugged-in (Section 4.3). Thus, in all our experiments we place 3DTRL after 4 Transformer layers, unless otherwise stated. The configuration of our DeiT-T, DeiT-S, and DeiT-B is identical to that mentioned in [52]. All our transformer models are trained for 50 and 300 epochs for CIFAR and ImageNet respectively. Further training details and hyper-parameters can be found in the supplementary material.

Table 1: Top 1 classification accuracy (%) on CIFAR-10, CIFAR-100, ImageNet-1K (IN-1K), and viewpoint-perturbed IN-1K (IN-1K-perturbed). We also report the number of parameters (#params) and computation (in MACs). Note that the MACs are reported w.r.t. IN-1K samples. All the models are trained from scratch.

Method	#params	MACs	CIFAR-10	CIFAR-100	IN-1K	IN-1K-perturbed
DeiT-T	5.72M	1.08G	74.1	51.3	73.4	61.3
+3DTRL	5.95M	1.10G	**78.8** (+4.7)	**53.7** (+2.4)	**73.6** (+0.2)	**64.6** (+3.3)
DeiT-S	22.1M	4.24G	77.2	54.6	79.4	71.1
+3DTRL	23.0M	4.33G	**80.7** (+3.5)	**61.5** (+6.9)	**79.7** (+0.3)	**72.7** (+1.6)
DeiT-B	86.6M	16.7G	76.6	51.9	81.0	70.6
+3DTRL	90.1M	17.2G	**82.8** (+6.2)	**61.8** (+9.9)	**81.2** (+0.2)	**74.7** (+4.1)

Figure 2: **Top**: original IN-1K samples. **Bottom**: viewpoint-perturbed IN-1K samples.

Results From Table 1, 3DTRL with all DeiT variants shows consistent performance improvement over the baseline on the CIFAR datasets, with only ∼2% computation overhead and ∼4% more parameters. Despite fewer training samples, 3DTRL significantly outperforms the baseline in CIFAR, showing the strong generalizability of 3DTRL when limited training data are available. We argue that multi-view data is not available in abundance, especially in domains with limited data (like in medical domain), thus 3DTRL with its ability to learn viewpoint-agnostic representations will be instrumental in such domains. We also find the performance improvement on ImageNet is less than that on CIFAR. This is because ImageNet has limited viewpoints in both training and validation splits, thus reducing the significance of performing geometric aware transformations for learning view agnostic representations. In contrast, CIFAR samples present more diverse camera viewpoints, so it is a more suitable dataset for testing the quality of learned viewpoint-agnostic representations.

Robustness on Viewpoint-perturbed Data In order to emphasize the need of learning viewpoint-agnostic representations, we further construct a *viewpoint*-perturbed ImageNet-1K validation set by applying random perspective transformations to images. Example images are shown in Figure 2. We note that perspective transformation on these static images is not equivalent to real viewpoint changes because the zero-paddings and interpolations do not happen in real-world cameras. Nonetheless, we believe it is a meaningful experiment as real image datasets with multi-view settings are lacking in the vision community. Table 1 shows that the performance improvement with 3DTRL on perturbed data is greater than that on original data. This demonstrates the robustness of 3DTRL for perspective transformed data, thanks to the learned viewpoint-agnostic representations.

4.2 Multi-view Video Alignment

Video alignment [16, 20, 36] is a task to learn a frame-to-frame mapping between video pairs with close semantic embeddings. In particular, we consider a *multi-view* setting that aligning videos captured from the same event but different viewpoints. Thus, video pairs from the same event are temporally synchronized.

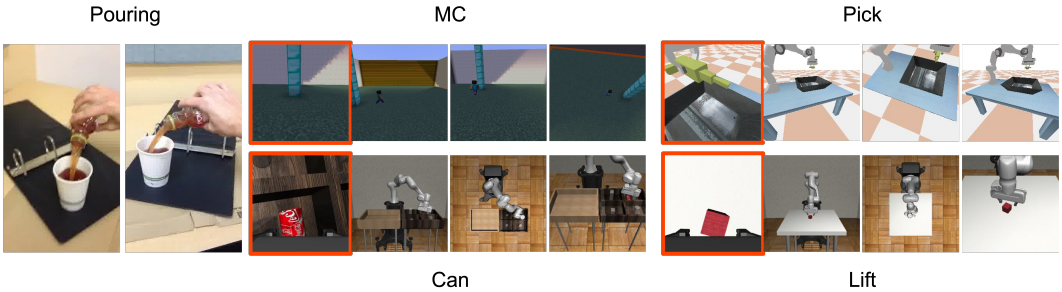
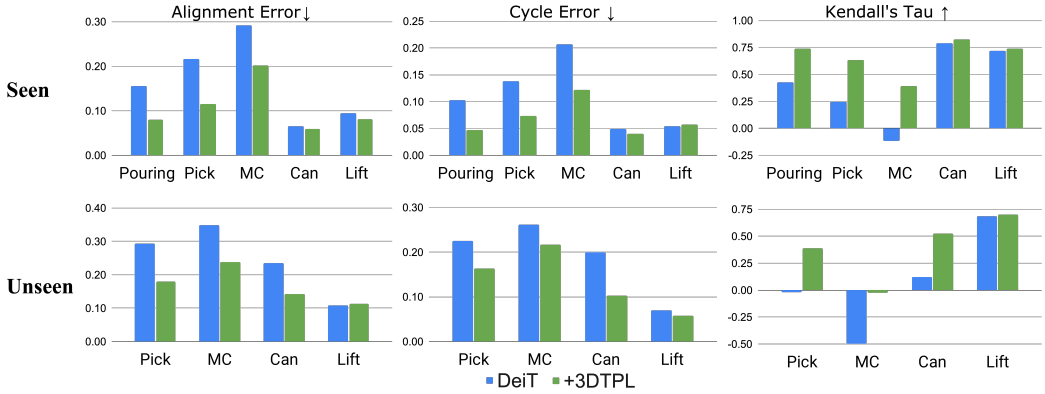


Figure 3: Examples video alignment datasets. Each dataset has synchronized videos of at least 2 viewpoints. All datasets except Pouring have one ego-centric view, highlighted in red boxes. More details are available in supplementary material.

Datasets We use 5 multi-view datasets from a wide range of environments: **Minecraft** (MC) – video game, **Pick**, **Can**, and **Lift** from robot simulators (PyBullet [7] and Robomimic [35]), and **Pouring** from real-world human actions [43]. Example video frames are provided in Figure 3. Each dataset contains synchronized videos from multiple cameras (viewpoints). There is one ego-centric camera per dataset except Pouring, which are continuously moving with the subject. These ego-centric videos make the alignment more challenging. Detail dataset statistics are available in supplementary.

Training We follow common video alignment methods [43] to train an encoder that outputs frame-wise embeddings. We still use DeiT [52] as a baseline model and apply 3DTRL to it, similar to image classification. During training, we use the time-contrastive loss [43] to encourage temporally closed embeddings to be similar while temporally far-away embeddings to be apart. Then, we obtain alignments via nearest-neighbor such that an embedding u_i from video 1 is being paired to its nearest neighbor v_j in video 2 in the embedding space. And similarly u_j from video 2 is paired with its nearest neighbor v_k in video 1. We use ImageNet-1K pre-trained weights for experiments on Pouring, but we train from scratch for other datasets considering that simulation environments are out of real-world distribution.

Figure 4: Results on video alignment in **Seen** and **Unseen** protocols. ↑ indicates a higher metric is better and ↓ is otherwise. Blue bars are for DeiT without 3DTPL and green bars are with 3DTPL. 3DTRL outperforms the baseline consistently in both settings. We note **Unseen** is not applicable to Pouring dataset because only two cameras are available in this dataset.



Evaluation We evaluate the alignment by three metrics: Alignment Error [43], Cycle Error, and Kendall's Tau [29]. Let the alignment pairs between two videos be (u_i, v_j) and (v_j, u_k). In brief, Alignment Error measures the temporal mismatching $|i - j|$ of (u_i, v_j). Cycle Error is based on cycle-consistency [16, 55], where two pairs (u_i, v_j) and (v_j, u_k) are called *consistent* when $i = k$. Thus, Cycle Error measures the inconsistency based on distance metric $|i - k|$. Kendall's Tau (τ) measures ordering in pairs. Given a pair of embeddings from video 1 (u_i, u_j) and their corresponding nearest neighbors from video 2 (v_p, v_q), the indices tuple (i, j, p, q) is *concordant* when $i < j$ and $p < q$ or $i > j$ and $p > q$. Otherwise the tuple is *discordant*. Kendall's Tau computes the ratio of concordant pairs and discordant pairs over all pairs of frames. Let N be the number of frames in a video, then the formal notations of the three metrics are:

$$\text{Alignment Error} = \mathbb{E}_i \frac{|i-j|}{N}; \ \text{Cycle Error} = \mathbb{E}_i \frac{|i-k|}{N}; \ \tau = \frac{\text{\# concordant pairs} - \text{\# discordant pairs}}{N(N-1)/2}. \tag{6}$$

We establish two evaluation protocols: (a) **Seen** and (b) **Unseen**. In **Seen**, we train and test models on videos from all cameras. However, in **Unseen**, we hold out several cameras for test, which is a representative scenario for validating the effectiveness of 3DTRL. Detail of experimental settings are provided in supplementary.

Results Figure 4 illustrates the evaluation results of 2 viewpoint settings over 5 datasets, compared to the DeiT baseline. 3DTRL outperforms the baseline consistently across all datasets. In particular, 3DTRL improves the baseline by a large margin in Pouring and MC, corroborating that 3DTRL adapts to diverse unseen viewpoints. The improvements of 3DTRL on Pick and MC also suggests the strong generalizability when learning from smaller datasets. With enough data (Lift and Can), 3DTRL still outperforms but the gap is relatively small. When evaluating in **Unseen** setting, both methods have performance drop. However, 3DTRL still outperforms in Pick, MC, and Can, which suggests the representations learned by 3DTRL are able to generalize over novel viewpoints. MC has the largest viewpoint diversity so it is hard to obtain reasonable align results in the unseen setting for both the models.

In Table 2, we further compare 3DTRL with previous methods on certain datasets. Note that Disentanglement [45] uses extra losses whereas others use time-contrastive loss only. We find that 3DTRL with only *single-frame* input is able to surpass the strong baselines set by models using extra losses [45] or multiple input frames [15]. We also vary the number of training videos in Pouring dataset, and results from Figure 3 show that 3DTRL benefits from more data. Meanwhile, 3DTRL can outperform the baseline while only using 22% of data the baseline used.

4.3 Ablation Studies

We conduct our ablation studies on image models mostly using *CIFAR* for image classification and *Pick* for multi-view video alignment.

Table 2: Video alignment results compared with SOTA methods. Values are alignment errors.

Method	Backbone	Input	Pouring	Pick	MC
TCN [43]	CNN	1 frame	0.180	0.273	0.286
Disentanglement [45]	CNN	1 frame	-	0.155	0.233
mfTCN [15]	3DCNN	8 frames	0.143	-	-
mfTCN [15]	3DCNN	32 frames	0.088	-	-
DeiT [52]	Transformer	1 frame	0.155	0.216	0.292
DeiT+3DTRL	Transformer	1 frame	**0.080**	**0.116**	**0.202**

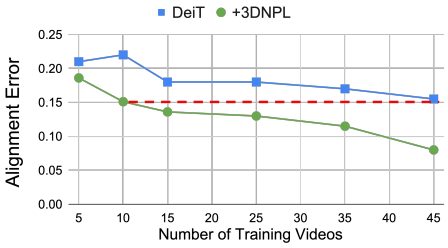


Table 3: Alignment error w.r.t # training videos. Red dashed line indicates 3DTRL using 10 videos outperforms DeiT using 45.

Table 4: Ablation study results. For CIFAR, we test on models based on tiny (T), small (S) and base (B) backbones (DeiT) and report accuracy(%). For Pick we only test base model and report alignment error.

Method	CIFAR-10 (T/S/B)	CIFAR-100 (T/S/B)	Pick
DeiT	74.1/ 77.2 / 76.6	51.3 / 54.6 / 51.9	0.216
DeiT + MLP	74.2 / 77.2 / 76.5	47.9 / 54.7 / 53.4	0.130
DeiT + 3DTRL	78.8 / 80.7 / 82.8	**53.7** / 61.5 / **61.8**	**0.116**
Depth Estimation $\rightarrow$ xyz Estimation	76.7 / 78.2 / 77.4	48.3 / 54.1 / 52.6	0.134
Embedding $\rightarrow$ Concat.	**80.7** / **83.7** / **84.9**	53.4 / **61.8** / 60.2	0.133

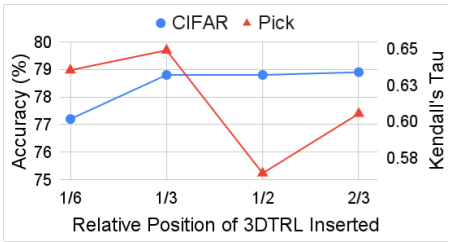


Figure 5: Results on inserting 3DTRL at different locations.

MLP vs. 3DTRL. 3DTRL is implemented by several MLPs with required geometric transforms in between. In this experiment, we replace 3DTRL with the similar number of fully-connected layers with residual connection, to have comparable parameters and computation as 3DTRL. Results are provided in Table 4. We find that MLP implementation is only comparable with the baseline performance despite the increase in parameters and computation. Thus, we confirm that the geometric transformations imposed on the token representations is the key to make 3DTRL effective.

Token Coordinates Estimation. In this ablation, we show how estimating only depth compared to estimating a set of 3 coordinates xyz differs in 3DTRL. We find at Line 3 and 4 Table 4 that depth estimation is better because it uses precise 2D image coordinates when recovering tokens in 3D space, whereas estimating xyz regresses for the 3D coordinates without any geometric constraints. Also, we find that estimating xyz hampers the performance in image classification task more than video alignment. This is because estimating 3D coordinates is harder when the training samples are in unconstrained scenarios. In contrast, video pairs in an alignment dataset share the same scene captured from different view angles, which facilitates the recovery of token in 3D space.

How to incorporate 3D positional information in Transformers? By default, we use Equation 5 to incorporate the 3D positional information in Transformer backbone through a learned positional embedding p^{3D}. In this experiment, we directly infuse the estimated 3D world coordinates p^{world} within the token representation by concatenating them across the channel axis. The $(m+3)$-d feature is then projected back to m-d by a MLP. We keep parameters and computation comparable to the default 3DTRL. We test this 3DTRL variant (Embedding $\rightarrow$Concat.) and results are presented in Table 4. We find that the concatenation variant outperforms the default variant of 3DTRL on CIFAR-10, but comparable and worse results in CIFAR-100 and Pick. This observation substantiates the instability of using raw 3D coordinates. In comparison, the use of 3D positional embedding is generalizable to more challenging and diverse scenarios.

Where should we have 3DTRL? We vary the location of 3DTRL to empirically study the optimal location of 3DTRL in a 12-layer DeiT-T backbone. In Figure 5 on the right, we find that inserting 3DTRL at the earlier layers (after 1/3 of the network) yields the best performance consistently on both datasets.

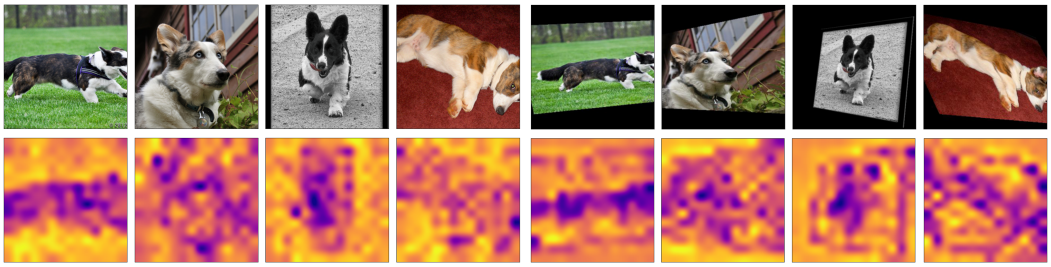


Figure 6: **Top**: ImageNet-1K validation samples (**left** half) and their viewpoint-perturbed versions (**right** half). **Bottom**: Estimated pseudo-depth map, interpolated from 16×16 to 224×224 for better understanding. Depth increases from blueish to yellowish region.

4.4 Qualitative Visualization of 3DTRL's Pseudo-depth Map

Although previous experiments have shown that 3DTRL is beneficial for different vision tasks, in this section, we visualize some qualitative results for a deeper understanding of 3DTRL's effectiveness. The qualitative results include visualization of estimated pseudo-depth maps which are from the key intermediate step of 3DTRL. For this analysis, we use an ImageNet-1K trained DeiT-T+3DTRL and its validation samples for visualization. In Figure 6, we observe a fine separation of the object-of-interest and the background in most of the pseudo-depth maps. Thus, these predicted pseudo-depth maps are sufficient to recover the tokens corresponding to object-of-interest in 3D space.

4.5 3DTRL for Video Representation Learning

Table 5: Results on action recognition on Smarthome (CV2 & CS protocols) and NTU (CV protocol). *Acc* indicates classification accuracy (%) and *mPA* indicates mean per-class accuracy. In methods using Kinetics-400 (K400) pre-training, TimeSformer backbone is always initialized with pre-trained weights, and 3DTRL w/, w/o K400 denotes 3DTRL is randomly initialized and is initialized from pre-trained weights respectively.

Method	**Smarthome** (CV2)		**Smarthome** (CS)		**NTU** (CV)
	Acc	*mPA*	*Acc*	*mPA*	*Acc*
TimeSformer [2]	59.4	27.5	75.7	56.1	86.4
+ 3DTRL	**62.9** (+3.5)	**34.0** (+6.5)	**76.1** (+0.4)	**57.0** (+0.9)	**87.9** (+1.5)
		Kinetics-400 pre-trained			
TimeSformer [2]	69.3	37.5	77.2	57.7	87.7
+ 3DTRL w/o K400	69.5 (+0.2)	39.2 (+1.7)	77.5 (+0.3)	58.9 (+1.2)	**88.8** (+1.1)
+ 3DTRL w/ K400	**71.9** (+2.6)	**41.7** (+4.2)	**77.8** (+0.6)	**61.0** (+2.3)	88.6 (+0.9)

In this section, we illustrate how 3DTRL can be adapted for video models. Since we aim at learning viewpoint-agnostic representations, as a natural choice we validate the effectiveness of 3DTRL on video datasets with multi-camera setups and cross-view evaluation. Consequently, we conduct our experiments on two multi-view action recognition datasets: **Toyota Smarthome** [10] (Smarthome) and **NTU-RGB+D** [44] (NTU) for the task of action classification. For evaluation on Smarthome, we follow Cross-View 2 (CV2) and Cross-Subject (CS) protocols proposed in [10], whereas on NTU, we follow Cross-View (CV) protocol proposed in [44]. In cross-view protocols, the model is trained on a set of cameras and tested on a different set of cameras. Similarly for cross-subject protocol, the model is trained and tested on different set of subjects. Note that for Smarthome dataset, we report the action classification accuracy on two metrics (i) classification accuracy (*Acc*) and (ii) mean per-class accuracy (*mPA*). More details on NTU and Smarthome datasets is provided in the supplementary material.

Network architecture & Training/Testing TimeSformer [2] is a straightforward extension of ViT [14] for videos which operates on spatio-temporal tokens from videos, so that 3DTRL can be easily deployed to TimeSformer as well. We use clips of size $8 \times 224 \times 224 \times 3$, with frames sampled at a rate of 1/32. We use a ViT-B encoder with patch size 16×16. Similar to our previous experimental settings, we place 3DTRL after 4 Transformer blocks in TimeSformer. We adopt

the training strategies mentioned in [2] and thus, train all the video models for 15 epochs. During inference, we sample one and 10 temporal clips from the entire video on NTU and Smarthome datasets respectively. We use 3 spatial crops (top-left, center, bottom-right) from each temporal clip and obtain the final prediction by averaging the scores for all the crops.

Results In Table 5, we present the action classification results on Smarthome and NTU datasets with 3DTRL plugged in TimeSformer. Firstly, we present the results for the models without any pre-training and then for the models with Kinetics-400 [28] (K400) pre-training.

3DTRL can easily take advantage of pre-trained weights because it does not change the relying backbone Transformer – just being added in between blocks. In Table 5, we present results for two fine-tuning scenarios: (a) 3DTRL w/o K400 and (b) 3DTRL w/ K400. For the first scenario (a), TimeSformer is initialized with K400 pre-training weights and leave the parameters in 3DTRL randomly initialized. Then in the fine-tuning stage, all the model parameters including that of 3DTRL is trained. In the second scenario (b), all parameters in TimeSformer and 3DTRL are pre-trained on K400 from scratch and fine-tuned on the respective datasets.

We find that all the variants of 3DTRL outperforms the baseline TimeSformer results. Our experiments show that although there is an improvement with 3DTRL compared to the baseline for different fine-tuning strategy, it is more significant when 3DTRL is pre-trained with K400. However, when large-scale training samples are available like in NTU, 3DTRL does not require K400 pre-training. To sum up, 3DTRL can be seen as a crucial ingredient for learning viewpoint-agnostic video representations.

5 Related Work

There has been a remarkable progress in visual understanding with the shift from the use of CNNs [4, 23, 31, 51] to visual Transformers [14]. Transformers have shown substantial improvements over CNNs in image [5, 14, 21, 27, 32, 52, 57] analysis and video [1, 2, 33, 39, 42] understanding tasks due to its flexibility in learning global relations among visual tokens. Studies also combine CNNs with Transformer architectures to leverage the pros in both the structures [8, 19, 34, 46]. In addition, Transformer has been shown to be effective in learning 3D representation [58]. However, these advancements in architecture-types have not addressed the issue of learning viewpoint-agnostic representation. Viewpoint-agnostic representation learning is drawing increasing attention in the vision community due to its wide range of downstream applications like 3D object-detection [41],video alignment [6, 16, 17], action recognition [47, 48], pose estimation [22, 50], robot learning [24, 26, 43, 45, 49], and other tasks.

There is a broad line of work towards directly utilizing 3D information like depth [22], pose [11, 12], and point clouds [38, 40], or in some cases deriving 3D structure from paired 2D inputs [54]. However, methods rely on the availability of multi-modal data which is hard to acquire are not scalable.

Consequently, other studies have focused on learning 3D perception of the input visual signal in order to generalize the learned representation to novel viewpoints. This is done by imposing explicit geometric transform operations in CNNs [3, 25, 37, 41, 56], without the requirement of any 3D supervision. In contrast to these existing works, our Transformer-based 3DTRL imposes geometric transformations on visual tokens to recover their representation in a 3D space. To the best of our knowledge, 3DTRL is the first of its kind to learn a 3D positional embedding associated with the visual tokens for viewpoint-agnostic representation learning in different image and video related tasks.

6 Conclusion

In this work, we have presented 3DTRL, a plug-and-play module for visual Transformer that leverages 3D geometric information to learn viewpoint-agnostic representations. Within 3DTRL, by pseudo-depth estimation and learned camera parameters, it manages to recover positional information of tokens in a 3D space. Through our extensive experiment, we confirm 3DTRL is generally effective in a variety of visual understanding tasks including image classification, multi-view video alignment, and cross-view action recognition, by adding minimum parameters and computation overhead.

References

[1] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021.

[2] G. Bertasius, H. Wang, and L. Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

[3] A.-Q. Cao and R. de Charette. Monoscene: Monocular 3d semantic scene completion. *arXiv preprint arXiv:2112.00726*, 2021.

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[5] C.-F. R. Chen et al. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[6] M. Chen, F. Wei, C. Li, and D. Cai. Frame-wise action representations for long videos via sequence contrastive learning. *arXiv preprint arXiv:2203.14957*, 2022.

[7] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[8] Z. Dai, H. Liu, Q. V. Le, and M. Tan. CoAtNet: Marrying convolution and attention for all data sizes. In *Proc. Adv. Neural Inform. Process. Syst.*, Dec. 2021.

[9] S. Das and M. S. Ryoo. Viewclr: Learning self-supervised video representation for unseen viewpoints. *CoRR*, abs/2112.03905, 2021. URL `https://arxiv.org/abs/2112.03905`.

[10] S. Das, R. Dai, M. Koperski, L. Minciullo, L. Garattoni, F. Bremond, and G. Francesca. Toyota smarthome: Real-world activities of daily living. In *ICCV*, 2019.

[11] S. Das, S. Sharma, R. Dai, F. Bremond, and M. Thonnat. Vpn: Learning video-pose embedding for activities of daily living. In *European Conference on Computer Vision*, pages 72–90. Springer, 2020.

[12] S. Das, R. Dai, D. Yang, and F. Bremond. Vpn++: Rethinking video-pose embeddings for understanding activities of daily living. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[15] D. Dwibedi, J. Tompson, C. Lynch, and P. Sermanet. Learning actionable representations from visual observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1577–1584. IEEE, 2018.

[16] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1801–1810, 2019.

[17] J. Gao, M. Chen, and C. Xu. Fine-grained temporal contrastive learning for weakly-supervised temporal action localization. *arXiv preprint arXiv:2203.16800*, 2022.

[18] J. J. Georgia Gkioxari, Jitendra Malik. Mesh r-cnn. *ICCV 2019*, 2019.

[19] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12259–12269, 2021.

[20] I. Hadji, K. G. Derpanis, and A. D. Jepson. Representation learning via global temporal alignment and cycle-consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11068–11077, 2021.

[21] K. Han et al. Transformer in transformer. *Advances in Neural Information Processing Systems*, 2021.

[22] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei. Towards viewpoint invariant 3d human pose estimation. In *European Conference on Computer Vision*, pages 160–177. Springer, 2016.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[24] K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn. Vision-based manipulators need to also see from their hands. *arXiv preprint arXiv:2203.12677*, 2022.

[25] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[26] R. Jangir, N. Hansen, S. Ghosal, M. Jain, and X. Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters*, 2022.

[27] K. Kahatapitiya and M. S. Ryoo. Swat: Spatial structure within and among tokens. *arXiv preprint arXiv:2111.13677*, 2021.

[28] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[29] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[30] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[31] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[32] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[33] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021.

[34] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[35] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[36] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[37] A. Piergiovanni and M. S. Ryoo. Recognizing actions in videos from unseen viewpoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4124–4132, June 2021.

[38] A. Piergiovanni, V. Casser, M. S. Ryoo, and A. Angelova. 4d-net for learned multi-modal alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15435–15445, 2021.

[39] K. Ranasinghe, M. Naseer, S. Khan, F. S. Khan, and M. Ryoo. Self-supervised video transformer. *arXiv preprint arXiv:2112.01514*, 2021.

[40] D. Robert, B. Vallet, and L. Landrieu. Learning multi-view aggregation in the wild for large-scale 3d semantic segmentation. *arXiv preprint arXiv:2204.07548*, 2022.

[41] D. Rukhovich, A. Vorontsova, and A. Konushin. Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2397–2406, 2022.

[42] M. S. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021.

[43] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[44] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[45] J. Shang and M. S. Ryoo. Self-supervised disentangled representation learning for third-person imitation learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 214–221, 2021. doi: 10.1109/IROS51168.2021.9636363.

[46] J. Shang and M. S. Ryoo. Starformer: Transformer with state-action-reward representations. *arXiv preprint arXiv:2110.06206*, 2021.

[47] G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari. Actor and observer: Joint modeling of first and third-person videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7396–7404, 2018.

[48] G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. *arXiv preprint arXiv:1804.09626*, 2018.

[49] B. C. Stadie, P. Abbeel, and I. Sutskever. Third-person imitation learning. *ArXiv*, abs/1703.01703, 2017.

[50] J. J. Sun, J. Zhao, L.-C. Chen, F. Schroff, H. Adam, and T. Liu. View-invariant probabilistic embedding for human pose. In *European Conference on Computer Vision*, pages 53–70. Springer, 2020.

[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.

[52] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou. Training data-efficient image transformers: distillation through attention. In *International Conference on Machine Learning*, volume 139, pages 10347–10357, July 2021.

[53] A. Vaswani et al. Attention is all you need. *Advances in neural information processing systems*, 2017.

[54] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[55] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2566–2576, 2019.

[56] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *Advances in neural information processing systems*, 29, 2016.

[57] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021.

[58] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021.

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]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (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, hyper-parameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (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]
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? [N/A]
   (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]