MVT: Multi-view Vision Transformer for 3D Object Recognition

Shuo Chen*
Tan Yu
Ping Li
Cognitive Computing Lab,
Baidu Research,
USA
{shanshuo1992,tanyuuynat,pingli98}@gmail.com

Abstract

Inspired by the great success achieved by CNN in image recognition, view-based methods applied CNNs to model the projected views for 3D object understanding and achieved excellent performance. Nevertheless, multi-view CNN models cannot model the communications between patches from different views, limiting its effectiveness in 3D object recognition. Inspired by the recent success gained by vision Transformer in image recognition, we propose a Multi-view Vision Transformer (MVT) for 3D object recognition. Since each patch feature in a Transformer block has a global reception field, it naturally achieves communications between patches from different views. Meanwhile, it takes much less inductive bias compared with its CNN counterparts. Considering both effectiveness and efficiency, we develop a global-local structure for our MVT. Our experiments on two public benchmarks, ModelNet40 and ModelNet10, demonstrate the competitive performance of our MVT.

1 Introduction

In the past decade, we have witnessed the great success achieved by convolutional neural network [13, 18] in image understanding. Inspired by its success in understanding 2D images, several works attempt to deploy CNN in 3D object understanding, achieving excellent performance. These methods can be coarsely divided into three groups: view-based methods [8, 11, 12, 17, 29, 32], volume-based methods [22, 23, 25, 36], and point-based methods [3, 26, 27]. Among them, view-based methods are closely related to 2D image understanding. View-based methods project a 3D object into multiple views and model each view through the model original used in modelling 2D images. Benefited from pre-training on the large-scale 2D image dataset such as ImageNet [5], they achieve competitive performance compared with their volume-based and point-based counterparts.

MVCNN [29] is the pioneering view-based method. It extracts each view features through a vanilla 2D CNN and aggregates the view features through sum-pooling. The following works [8, 11, 12, 17, 32, 34] seek to find a more effective way to aggregate the view features. Specifically, RCPCNN [32] and GVCNN [8] group views into multiple sets and conduct pooling within each set. Seqviews2seqlabels [12] and 3D2SeqViews [11] model the

*Work performed during internship. Currently at University of Amsterdam.

view order through recurrent neural network. View-GCN [34] models the view-based relations through graph convolution network. MHBN [39] and MVLADN [40] observe the limitations of view-based pooling, formulates the view-based 3D object recognition into a set-to-set matching problem, and investigates in patch-level pooling. Nevertheless, view-based pooling and patch-based pooling methods only fuse the visual features from different views in the last pooling layer. There are no interactions between visual features from different views in previous layers. This configuration leads to the fact that a patch can only have a local perception field and fails to perceive patches in other views. Relation Network [38] enhances each patch feature by patches from all views, achieving better performance than the above-mentioned view-based pooling and patch-based pooling methods.

In this work, inspired by the great success achieved by visual Transformer [7, 30], we propose a multi-view visual Transformer (MVT) to empower each patch to have the global reception field to perceive the visual content of all views from a 3D object. It adopts a pure-Transformer architecture and thus takes much less inductive bias compared with its CNN counterparts [7]. Considering the total number of patches is largely due to multiple projected views, simply concatenating all patches will generate an extremely long sequence, leading to an expensive computational cost. Taking both effectiveness and efficiency into consideration, we devise a local-global structure, as visualized in Figure 1. In the local Transformer encoder, we adopt Transformer to process the patches within each view individually. In the global Transformer encoder, we merge patch features from all views and feed them together into Transformer layers for the global reception field. Using such a simple and elegant architecture, we achieve state-of-the-art recognition accuracy on public benchmarks, including ModelNet40 and ModelNet10.

2 Related Works

3D object recognition. Existing mainstream 3D object recognition methods can be categorized into three groups: volume-based methods [22, 23, 25, 36], point-based methods [3, 26, 27] and view-based methods [8, 11, 12, 17, 32, 34],. Among them, volume-based methods quantize the 3D object into regular voxels, and conduct 3D convolutions on voxels. Nevertheless, 3D convolution is computationally expensive when the resolution is high. For satisfactory efficiency, volume-based methods normally conduct low-resolution quantization, inevitably leading to information loss. In parallel, point-based methods directly model the cloud of points, efficiently achieving competitive performance. View-based methods project a 3D object into multiple 2D views. They model each view through the vision backbone for image understanding to obtain view features or patch features. Our work can be categorized into view-based methods. Thus, we mainly review view-based methods here.

MVCNN [29] is one of the earliest work exploiting convolutional neural network (CNN) for modelling multiple views. It aggregates the view features from CNN through max-pooling. MVCNN-MultiRes [25] exploits views projected from multi-resolution settings, boosting the recognition accuracy. Pairwise [16] decomposes the sequence of projected views into several pairs and models the pairs through CNN. GIFT [2] represents each 3D object by a set of view features and determines the similarity between two 3D objects by matching two sets of view features through a devised matching kernel. RotationNet [17] takes the viewpoint of each project into consideration, and treats viewpoints as latent variables to boost the recognition performance. RCPCNN [32] groups views into multiple sets and concatenates the set features as the 3D object representation. GVCNN [8] also groups views into

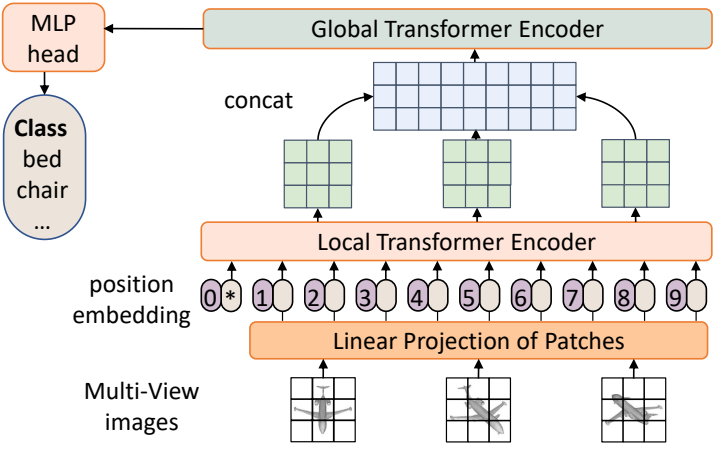

Figure 1: The architecture of the proposed multi-view vision Transformer (MVT). Each view is split into non-overlap patches. Each patch is projected into an embedding vector, which is added to a position embedding. Patch embedding vectors from each view are feed into the local Transformer encoder for communications between patches within the view. Then the outputs of the local Transformer for all views are concatenated into a global set, which is fed into the global Transformer encoder for communications between patches from different views. After that, the set of attended patch features from the output of the global Transformer encoder are sum-pooled into a global representation for the 3D object. We finally use the MLP head for classification.

multiple sets. It adaptively assigns a higher weight to the group containing crucial visual content to suppress the noise. Seqviews2seqlabels [12] and 3D2SeqViews [11] exploit the view order besides visual content through recurrent neural network. View-GCN [34] models the relations between views by a graph convolution network. MHBN [39] and MVLADN [40] investigate in pooling patch-level features to generate the 3D object recognition. Relation Network [38] enhances each patch feature by patches from all views through a reinforcement block plugged in the rear of the network. Our method has a similar spirit, but needs much less inductive bias and only takes a standard Transformer to achieve the communications between patches of different views.

Vision Transformer. Inspired by the great success achieved by Transformers [31] in natural language processing, vision Transformer (ViT) [7] is proposed. It crops an image into multiple non-overlap patches and feeds the cropped patches into a stack of Transformer layers. Compared with CNN models, each patch in ViT has a global reception field. Meanwhile, ViT has much less image-specific inductive bias than CNN models. By pre-training on huge-scale datasets, ViT has achieved comparable accuracy compared with its CNN counterparts. DeiT [30] proposes a data-efficient approach using an improved optimizer, more advanced data augmentation and training tricks. PVT [33] and PiT [14] bring back inductive bias in CNN and exploit the pyramid structure to progressively shrink the spatial size. T2T [41] and TNT [10] focus on improving the effectiveness of modeling local structure within patches. Swin [20] and Twin [4] exploit the locality and sparsity to achieve a better trade-off between effectiveness and efficiency. CvT [35] and Container [9] exploit a hybrid structure combining Transformer and convolution. Different from above-mentioned methods which exploiting Transformer for 2D image understanding, we investigate its effectiveness in 3D object recognition.

3 Preliminary

We briefly introduce the structure of Transformer [31] block as visualized in Figure 2. It consists of two layer-normalization (LN) layers, a multi-head self-attention (MSA) module and a multi-layer perceptron (MLP) module. Below we introduce them in details.

Multi-head self-attention (MSA) module. Let us denote the inputs by $\mathbf{X} \in \mathbb{R}^{N\times D}$ where N is the number of input vectors and D is the dimension of each input vector. MSA maps $\mathbf{X}$ into the queries $\mathbf{Q}^{N\times D_Q}$, keys $\mathbf{K} \in \mathbb{R}^{N\times D_K}$ and values $\mathbf{V} \in \mathbb{R}^{N\times D_V}$ through three fully-connected layers, where $D_Q = D_K = D_V$. Then it splits $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ into M heads:

$$\mathbf{Q} \to [\mathbf{Q}_1, \cdots, \mathbf{Q}_M],\ \mathbf{K} \to [\mathbf{K}_1, \cdots, \mathbf{K}_M],\ \mathbf{V} \to [\mathbf{V}_1, \cdots, \mathbf{V}_M],$$

where

$$\mathbf{Q}_m \in \mathbb{R}^{N\times \frac{D_Q}{M}},\ \mathbf{K}_m \in \mathbb{R}^{N\times \frac{D_K}{M}},\ \mathbf{V}_m \in \mathbb{R}^{N\times \frac{D_V}{M}},\ m = 1, \ldots, M.$$

Then the self-attention operation is conducted on each query-key-value triplet $\{\mathbf{Q}_m, \mathbf{K}_m, \mathbf{V}_m\}$ $(m = 1, \ldots, M)$ and generates the attended features:

$$\mathbf{Y}_m = \text{softmax}(\frac{\mathbf{Q}_m \mathbf{K}_m^\top}{\sqrt{D_k}})\mathbf{V}_m,\ m = 1, \ldots, M.$$

The attended features from each head are concatenated to obtain the final output of the self-attention module:

$$\mathbf{Y} \leftarrow [\mathbf{Y}_1, \cdots, \mathbf{Y}_M] \in \mathbb{R}^{N\times D_V}.$$

Layer Normalization (LN) [1] is widely used in Transformer-based architecture for training stability. Given a D-dimension feature vector $\mathbf{x} = [x_1, \cdots, x_D]$, it computes the mean μ and the standard deviation ε by

$$\mu = \frac{1}{D}\sum_{i=1}^{D} x_i, \ \varepsilon = \sqrt{\frac{1}{D}\sum_{i=1}^{D}(x_i - \mu)^2}.$$

Then a linear operation is conducted on each element of $\mathbf{x}$:

$$\hat{x}_i = \gamma \frac{x_i - \mu}{\varepsilon} + \beta, \tag{1}$$

where β and γ are learnable parameters for affine transform.

Multi-layer perceptron (MLP). The MLP is normally plugged after self-attention module to operate on each input separately [31]. It consists of two fully-connected layers with a bottleneck structure and an activation layer for for non-linearity. Specifically, for each feature $\mathbf{x} \in \mathbb{R}^D$, MLP enhances $\mathbf{x}$ by

$$\text{MLP}(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{2}$$

where $\mathbf{W}_1 \in \mathbb{R}^{D\times rD}$ and $\mathbf{b}_1 \in \mathbb{R}^{rD}$ are weights of the first fully-connected layer, which increase the feature dimension from D to rD and $r > 1$ is termed as expansion ratio. Meanwhile, $\mathbf{W}_2 \in \mathbb{R}^{rD\times D}$ and $\mathbf{b}_2 \in \mathbb{R}^D$ are weights of the second fully-connected layer, which decreases the feature dimension from rD back to D.

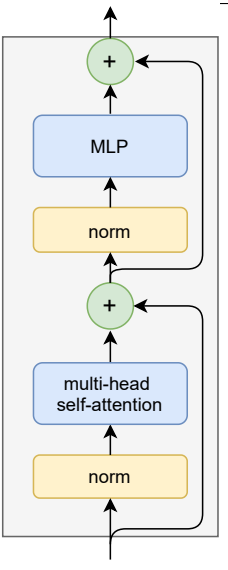

Figure 2: The structure of a Transformer block. It consists of layer-normalization (norm) layers, a multi-layer perceptron (MLP) module and a multi-head self-attention module.

4 Method

In this section, we introduce the proposed multi-view visual Transformer. We first clarify the input of the model and then illustrate the details of the model.

4.1 Input

For each 3D object, we project it into L views $\{V^1, \cdots, V^L\}$. Each view V^j $(j = 1, \ldots, L)$ is of $W \times H \times 3$ size. For each view, we crop it into $w \times h$ non-overlap patches and each patch is of $p \times p \times 3$ size. Each patch is unfolded into a vector $\mathbf{p} \in \mathbb{R}^{3p^2}$. We denote the i-th patch in the view V^j by $\mathbf{p}_i^j$, where $i \in [1, wh]$ and $j \in [1, L]$. For each $\mathbf{p}_i^j$, we map it into a D-dimension vector through a fully-connected layer and obtain

$$\mathbf{x}_i^j \leftarrow \mathbf{W}_0\mathbf{p}_i^j, \ i = 1, \ldots, wh, \ j = 1, \ldots, L, \tag{3}$$

where $\mathbf{W}_0 \in \mathbb{R}^{D\times 3p^2}$ is the weight matrix. Meanwhile, a position embedding $\mathbf{p}_i$ is learned for each spatial location $i \in [1, wh]$. Then the patch feature is obtained by summing up its visual feature and position embedding:

$$\mathbf{z}_i^j \leftarrow \mathbf{x}_i^j + \mathbf{p}_i. \tag{4}$$

Note that, $\mathbf{p}_i$ is only dependent on the spatial location (i) and is shared among different views. Like BERT's `[class]` token [6], for each view V^j, we additionally devise a special token, $\mathbf{z}_0^j$, which is a learnable embedding whose state at the output of the Transformer encoder serves as the view representation. We merge the special token $\mathbf{z}_0^j$ and the patch features $\{\mathbf{z}_i^j\}_{i=1}^{wh}$ into a matrix $\mathbf{Z}^j$ defined as

$$\mathbf{Z}^j = [\mathbf{z}_0^j, \mathbf{z}_1^j, \cdots, \mathbf{z}_{wh}^j] \in \mathbb{R}^{D\times(wh+1)}. \tag{5}$$

$\{\mathbf{Z}^j\}_{j=1}^L$ are the inputs of our multi-view visual Transformer.

4.2 Multi-view visual Transformer

The proposed multi-view visual Transformer consists of two parts. The first part processes patches in each view, individually. It generates the low-level features for patches in each view. We term the first part as local Transformer layers. The second part takes the low-level patch features from the first part as input. It merges patches features from all views in a set and feeds the merged set into a stack of Transformer layers for empowering each patch of each view to have a global reception field. We term these layers exploiting global visual content as global Transformer layers.

Local Transformer blocks. They process the patch features from each view, individually. Let us denote the number of local Transformer blocks as S. The feature matrix $\mathbf{Z}^j$ from each view V^j goes through S blocks sequentially. The input of the s-th local Transformer block is denoted by $\mathbf{Z}_{s-1}^j$ and the output by $\mathbf{Z}_s^j$. In this case, the input of the first block $\mathbf{Z}_0^j$ is just the patch feature set $\mathbf{Z}^j$ defined in Eq (5). Meanwhile, we denote the layers in the s-th local Transformer block by $\{\mathrm{MSA}_s^l, \mathrm{MLP}_s^l, \mathrm{LN}_{s,1}^l, \mathrm{LN}_{s,2}^l\}$. For the s-th local Transformer block, it conducts the following operation:

$$\begin{aligned}
\mathbf{Z}_s^j &\leftarrow \mathbf{Z}_{s-1}^j + \mathrm{MSA}_s^l(\mathrm{LN}_{s,1}^l(\mathbf{Z}_{s-1}^j)), \quad j = 1\ldots L, \\
\mathbf{Z}_s^j &\leftarrow \mathbf{Z}_s^j + \mathrm{MLP}_s^l(\mathrm{LN}_{s,2}^l(\mathbf{Z}_s^j)), \quad j = 1\ldots L.
\end{aligned} \tag{6}$$

After S local Transformer blocks, we obtain the output, $\mathbf{Z}_S^j$ for each view V^j.

Global Transformer blocks. They process the patch features from each view, jointly. At first, the output of the last local Transformer block for all views, $\{\mathbf{Z}_S^j\}_{j=1}^L$ are concatenated into a global matrix:

$$\mathbf{M} = [\mathbf{Z}_S^1, \cdots, \mathbf{Z}_S^L] \in \mathbb{R}^{D\times Lwh}. \tag{7}$$

We denote the layers in the t-th global Transformer block by $\{\mathrm{MSA}_t^g, \mathrm{MLP}_t^g, \mathrm{LN}_{t,1}^g, \mathrm{LN}_{t,2}^g\}$. The input of the t-th local Transformer block is denoted by $\mathbf{M}_{t-1}$ and the output is denoted by $\mathbf{M}_t$. For the t-the global Transformer block, it conducts the following operation:

$$\begin{aligned}
\mathbf{M}_t &\leftarrow \mathbf{M}_{t-1} + \mathrm{MSA}_t^g(\mathrm{LN}_{t,1}^g(\mathbf{M}_{t-1})), \\
\mathbf{M}_t &\leftarrow \mathbf{M}_t + \mathrm{MLP}_t^g(\mathrm{LN}_{t,2}^g(\mathbf{M}_t)).
\end{aligned} \tag{8}$$

After T global Transformer blocks, we obtain the output, $\mathbf{M}_T$:

$$\mathbf{M}_T = [\mathbf{m}_0^1, \mathbf{m}_1^1, \cdots, \mathbf{m}_{wh}^1, \mathbf{m}_0^2, \mathbf{m}_1^2, \cdots, \mathbf{m}_{wh}^2, \cdots, \mathbf{m}_0^L, \mathbf{m}_1^L, \cdots, \mathbf{m}_{wh}^L] \in \mathbb{R}^{D\times L(wh+1)}, \tag{9}$$

where $\mathbf{m}_0^j$ denotes the attended special token for the view V^j. We conduct sum-pooling on $\{\mathbf{m}_0^j\}_{j=1}^L$ and obtain the global representation for the 3D object:

$$\mathbf{m} = \frac{1}{L}\sum_{j=1}^{L}\mathbf{m}_0^j. \tag{10}$$

Then $\mathbf{m}$ is fed into a fully-connected layer for classification.

5 Experiments

Datasets. To evaluate the proposed methods, wer perform experiments on ModelNet40 and ModelNet10 [36]. ModelNet40 consists of 12,311 3D CAD models from 40 categories. Among them, 9,843 models are for training and 2,468 models are for testing. ModelNet10 is a subset of ModelNet40 and have 10 categories. We use two settings for generating views from the 3D object. The 12-view setting follows the setup in [32] and the 20-view setting follows the manner in [17].

	hidden dimension	# heads	# local blocks	# global blocks
tiny	192	3	8	4
small	384	6	8	4

Table 1: The details of different settings.

Implementation. The architecture of our MVT follows DeiT [30]. We also attempt several configurations including the tiny and small models. The details of different configurations are summarized in Table 1.

Training details. The training follows the settings in DeiT [30]. To be specific, we use AadmW [21] as the optimizer, with an initial learning rate of 0.001, β_1=0.9, β_2=0.98. Our model is implemented in PyTorch [24]. The model is trained with mixed precision on 2 NVIDIA V100 GPUs. We train 300 epochs for training from scratch and 100 epochs for fine-tuning on a pretrained model. The pretrained models are trained on ImageNet1K dataset [5], provided by Touvron *et al*. [30].

5.1 The influence of the global Transformer blocks

When the number of global Transformer blocks is 0, it is equivalent to replacing the CNN in MVCNN [29] by a vision Transformer. We term this configuration as the local baseline. In this baseline setting, each patch can only communicate with patches from the same view. When we replace the local transformer block with the global transformer block, the patch embedding from local transformer are feed into global transformer to interrelate with patches from other views.

Accuracy. To investigate how the number of global transformer blocks and local transformer blocks influence the performance, we use the ModelNet10 as a test bed to ablate. The results are shown in Table 2. When training tiny model from scratch, we can see that when we add global transformer blocks, no matter how deep, the performance improves compared to the local baseline. Training small model without fine-tuned on pre-trained model, including

	tiny						small					
local blocks	12	11	10	8	4	0	12	11	10	8	4	0
global blocks	0	1	2	4	8	12	0	1	2	4	8	12
w/o pre-train	90.42	91.02	91.30	92.35	92.07	91.13	92.57	92.35	92.73	93.12	92.79	92.35
w/ pre-train	94.55	94.16	94.38	94.82	94.82	94.66	94.77	94.71	95.10	95.21	95.04	95.21
GPU memory (M)	1523	1528	2114	2548	3394	4289	3072	3817	4242	5095	6843	8586
time/epoch (s)	15	17	17	20	23	26	24	25	27	33	40	49
throughput (obj/s)	53	50	47	47	36	18	25	23	23	19	16	15

Table 2: Evaluation of our method with different numbers of block layers on ModelNet10 dataset, where w/o pre-train denotes training from scratch and w/ pre-train means fine-tuning from a pre-trained model. The view number is fixed as 6. We set the batch size (the number of 3D objects per batch) as 8 when testing the GPU memory cost. The inference throughput is measured as the number of 3D objects processed per second on an NVIDIA TITAN X Pascal GPU. Setting 4 global transformer blocks strikes a good balance between interrelate intra-view and inter-view on patches.

global transformer blocks has better performance at most cases. This proves that adding global Transformer blocks to communicate patches from different views helps the classification performance. The local baseline achieves 90.42% using tiny model and and 93.12% using small model. When including 4 global Transformer blocks with tiny model, the performance achieves highest with accuracy of 92.35% compared to 90.42% of local baseline, compared to 92.57% of local baseline. However, increasing the number of global Transformer blocks does not always guarantee to better performance. When we have more than 4 global Transformer blocks, the accuracy declines. Less local Transformer blocks, leading to less layers where patches only attend to intra-view patches, could cause such decrease. We suppose intra-view attention for low layers is essential. If we alter all local transformer blocks with global transformer blocks, there is no layer to restrict the patches interrelate with other patches only within its view. This is another setting we call global baseline. However, when fine-tuned on pre-trained models, more global Transformer blocks does not always leads to a better performance.

To investigate the choice of the number of global Transformer blocks, we keep fixed local Transformer blocks while increase the global Transformer blocks from 0 to 6 gradually. Table 3 displays the accuracy on the ModelNet10 test set with different number of global Transformer blocks. We observe that more global Transformer blocks does not always lead to a better performance. The accuracy of 5 and 6 global Transformer blocks is lower than of 4. More global Transformer blocks have more trainable parameters. From Table 2 and Table 3 we conclude that 4 global transformer blocks setting is a good trade off and we set as the default.

global	0	1	2	3	4	5	6
accuracy	90.42	91.46	91.57	92.02	**92.35**	91.85	91.96

Table 3: Evaluating the number of global Transformer blocks with fixed local Transformer blocks.

Efficiency. Meanwhile, in the last rows of Table 2, we show the GPU memory cost per batch and the time cost per epoch of different settings. We include the inference speed

in Table 2 as well. It is shown that when we use more global blocks, the GPU memory and the training time increase accordingly. At the same time the inference speed decrease correspondingly. Specifically, using the small model, when the number of global blocks increases from 0 to 12, the GPU memory cost increases from 3072M to 8586M, the training time cost per epoch increases from 24 seconds to 49 seconds, and the throughput decreases from 25 objects/second to 15 objects/second. Considering both effectiveness and efficiency, we set the number of local Transformer blocks as 8 and that of global Transformer block as 4, by default.

Tiny vs Small. From Table 2 we can also see the performance improvement of using a larger model. The small model has 6 heads with hidden dimension of 384. Both the number of heads and dimensions are double to the tiny model. This leads to a better performance. Take 4 global Transformer blocks as example, the accuracy increases from 92.35% (tiny without pre-train) to 93.12% (small without pre-train).

5.2 The influence of the number of projected views

Accuracy. We evaluate the effect of the number of views on the average instance accuracy of our method on the ModelNet10 dataset. As shown in Table 4, more projected views leads to better classification accuracy. To be specific, using a single view, small model with pre-training only achieves a 89.98% recognition accuracy, whereas it achieves a 95.26% recognition accuracy using 12 views. This is expected since more views will provide more visual information for a 3D object and benefits the 3D object recognition.

Efficiency. We report the number of views on the GPU memory consumption and time cost per epoch in the last two rows of Table 4. As shown in the table, using more views leads to more computational cost and GPU memory cost. In detail, using the small model, the GPU memory per batch increases from 1010M to 10700M when the number of views increases from 1 to 12. We recommend to use only 3 projected views when the computing resources are limited since it has achieved a excellent accuracy. Meanwhile, we suggest to use 12 projected views when computing resources are abundant.

	tiny				small			
views	1	3	6	12	1	3	6	12
w/o pre-train	82.54	89.70	92.35	92.47	85.19	92.35	93.12	93.13
w/ pre-train	90.64	94.00	94.82	95.01	89.98	94.82	95.12	95.26
GPU memory (M)	365	1457	2548	5455	1010	2354	5095	10700
time/epoch (s)	5	11	20	33	25	14	33	68

Table 4: Evaluation of our method with different view numbers on ModelNet10 datasets.

5.3 The influence of the class token

In the current settings, we feed the attended class token feature in the output of the last Transformer block to the classifier to obtain the recognition result. An alternative choice is to average-pool all attended patch features in the output of the last Transformer block to generate a global representation for classification. We investigate the effectiveness of

Method	Views	ModelNet40	ModelNet10
Volume-based methods			
3DShapeNets [36]	-	77.0	83.5
VoxNet [22]	-	83.0	92.0
Volumetric CNN [25]	-	89.9	-
3D-A-Nets [28]	-	90.5	-
LP-3DCNN [19]	-	92.1	-
Point-based methods			
PointNet [26]	-	89.2	-
PointNet++ [27]	-	91.9	-
3DmFV-Net [3]	-	91.6	95.2
DeepCCFV [15]	-	92.5	-
View-based methods			
MVCNN [29]	80	90.1	-
RotationNet [17]	12	91.0	94.0
RotationNet [17]	20	97.4	98.5
Relation Network [38]	12	94.3	95.3
3D2SeqViews [11]	12	93.4	94.7
SeqViews2SeqLabels [12]	12	93.4	94.8
GVCNN [8]	12	93.1	-
CARNet [37]	12	95.2	95.8
CARNet [37]	20	**97.7**	99.0
MVT-small (Ours)	12	94.4	95.3
MVT-small (Ours)	20	97.5	**99.3**

Table 5: Comparison with the present state-of-the-art methods on ModelNet40 dataset.

leveraging class token compared to average-pooling patch features. The tiny model is trained from scratch on ModelNet10 with 6 views. In Table 6, we show the experimental results. The improved performance shows that the attended class token achieves a better recognition performance than its counterpart using the global feature obtained from average-pooling the attended patch features.

	avg_pool	class_token
accuracy	91.45%	92.35%

Table 6: Comparisons between the global feature from average pooling the attended patch features and the attend class token feature.

5.4 Comparsions with state-of-the-art methods

We compare with three groups of methods including volume-based methods, point-based methods and view-based methods in Table 5. The first part of Table 5 reports the performance of volume-based methods including 3DShapeNets [36], VoxNet [22], Volumetric CNN [25],

3D-A-Nets [28], and LP-3DCNN [19]. As shown in the table, the recognition accuracy of these volume-based methods are not competitive compared with view-based methods.

Then we compare with point-based methods including PointNet [26], PointNet++ [27], 3DmFV-Net [3], and DeepCCFV [15]. Compared with volume-based methods, point-based methods achieve considerably higher recognition accuracy. To be specific, PointNet++ [27] achieves 91.9 recognition accuracy. It significantly outperforms the best volume-based method in Table 5, 3D-A-Nets [28], with only 90.5 recognition accuracy. But the point-based methods are still not as competitive as their view-based counterparts.

At last, we compare with view-based methods including MVCNN [29], RotationNet [17], 3D2SeqViews [11], SeqViews2SeqLabels [12], Relation Network [38] and CARNet [37] on both 12-view and 20-view settings. With more views, the performance achieved by using 20-view settings is normally better than 12-view settings. Compared with these methods, our MVT-small model achieves competitive performance. Specifically, on ModelNet10 dataset, using 20-view settings, we achieve the highest recognition accuracy. It is worth noting that the architecture of our MVT-small is conceptually simple with less hand-designed components than the compared methods such as Relation Network [38] and CARNet [37].

6 Conclusion

In this paper, we propose a multi-view vision Transformer (MVT) for effective 3D object recognition. Considering the efficiency, we design our MVT in a local-global structure. The global Transformer layers empower each patch to communicate with the patches from all views, overcoming the limitations of existing CNN-based models with a local reception field on patches from the same view. Albeit the proposed MVT is in a very conceptually simple structure, it has achieved state-of-the-art recognition performance on public benchmarks including ModelNet40 and ModelNet10 datasets.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Song Bai, Xiang Bai, Zhichao Zhou, Zhaoxiang Zhang, and Longin Jan Latecki. Gift: A real-time and scalable 3d shape search engine. In *CVPR*, 2016.

[3] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *RA-L*, 2018.

[4] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv preprint arXiv:2104.13840*, 2021.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[8] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *CVPR*, 2018.

[9] Peng Gao, Jiasen Lu, Hongsheng Li, Roozbeh Mottaghi, and Aniruddha Kembhavi. Container: Context aggregation network. *arXiv preprint arXiv:2106.01401*, 2021.

[10] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

[11] Zhizhong Han, Honglei Lu, Zhenbao Liu, Chi-Man Vong, Yu-Shen Liu, Matthias Zwicker, Junwei Han, and C. L. Philip Chen. 3d2seqviews: Aggregating sequential views for 3d global feature learning by CNN with hierarchical attention aggregation. *TIP*, 2019.

[12] Zhizhong Han, Mingyang Shang, Zhenbao Liu, Chi-Man Vong, Yu-Shen Liu, Matthias Zwicker, Junwei Han, and C. L. Philip Chen. Seqviews2seqlabels: Learning 3d global features via aggregating sequential views by RNN with attention. *TIP*, 2019.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[14] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv: 2103.16302*, 2021.

[15] Zhengyue Huang, Zhehui Zhao, Hengguang Zhou, Xibin Zhao, and Yue Gao. DeepCCFV: Camera Constraint-Free Multi-View Convolutional Neural Network for 3D Object Retrieval. In *AAAI*, 2019.

[16] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, 2016.

[17] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *CVPR*, 2018.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[19] Sudhakar Kumawat and Shanmuganathan Raman. Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks. In *CVPR*, 2019.

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[22] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015.

[23] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. Vv-net: Voxel vae net with group convolutions for point cloud segmentation. In *ICCV*, 2019.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPS Workshops*, 2017.

[25] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, 2016.

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[28] Mengwei Ren, Liang Niu, and Yi Fang. 3d-a-nets: 3d deep dense descriptor for volumetric shapes with adversarial networks. *arXiv preprint arXiv:1711.10108*, 2017.

[29] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015.

[30] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[32] Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. In *BMVC*, 2017.

[33] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021.

[34] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *CVPR*, 2020.

[35] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

[36] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[37] Yong Xu, Chaoda Zheng, Ruotao Xu, Yuhui Quan, and Haibin Ling. Multi-view 3d shape recognition via correspondence-aware deep learning. *TIP*, 2021.

[38] Ze Yang and Liwei Wang. Learning relationships for multi-view 3d object recognition. In *ICCV*, 2019.

[39] Tan Yu, Jingjing Meng, and Junsong Yuan. Multi-view harmonized bilinear network for 3d object recognition. In *CVPR*, 2018.

[40] Tan Yu, Jingjing Meng, Ming Yang, and Junsong Yuan. 3d object representation learning: A set-to-set matching perspective. *TIP*, 2021.

[41] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token ViT: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.