This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Learning to Select Views for Efficient Multi-View Understanding

Yunzhong Hou, Stephen Gould, and Liang Zheng
Australian National University
{firstname.lastname}@anu.edu.au

Abstract

Multiple camera view (multi-view) setups have proven useful in many computer vision applications. However, the high computational cost associated with multiple views creates a significant challenge for end devices with limited computational resources. In modern CPU, pipelining breaks a longer job into steps and enables parallelism over sequential steps from multiple jobs. Inspired by this, we study selective view pipelining for efficient multi-view understanding, which breaks computation of multiple views into steps, and only computes the most helpful views/steps in a parallel manner for the best efficiency. To this end, we use reinforcement learning to learn a very light view selection module that analyzes the target object or scenario from initial views and selects the next-best-view for recognition or detection for pipeline computation. Experimental results on multi-view classification and detection tasks show that our approach achieves promising performance while using only 2 or 3 out of N available views, significantly reducing computational costs while maintaining parallelism over GPU through selective view pipelining[1].

1. Introduction

Multiple camera views (multi-view) are popular in computer vision systems for their ability to address challenges such as occlusions, ambiguities, and limited field-of-view (FoV) coverage. Tasks like classification [30, 37] and detection [6, 17] have shown significant benefits from using multiple cameras. With reduced hardware cost and easy deployment, real-world products now include more cameras at larger scales. However, the use of multiple cameras comes at a high computational cost, which can be a significant challenge for end devices with limited computational resources, especially with higher image resolutions and deeper neural network backbones. Limiting image resolution or using lighter networks [19, 27] are current options to reduce computation, but they may impede the progress in camera sensors or neural network architecture.

[1]Code available at https://github.com/hou-yz/MVSelect.

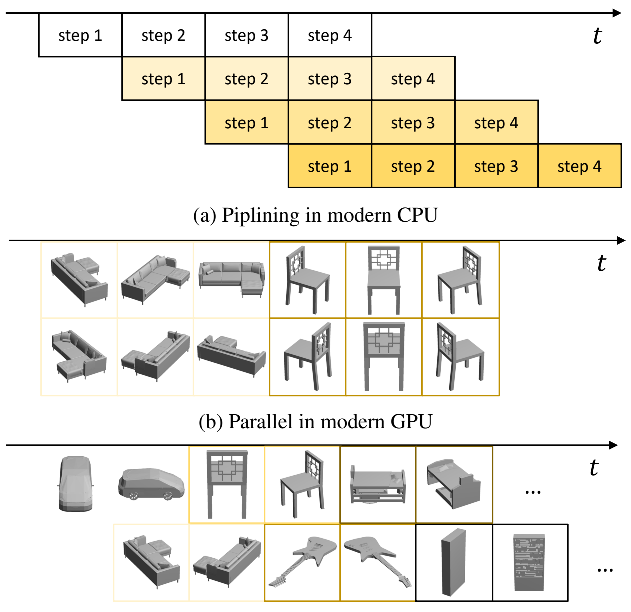

step 1
step 2
step 3
step 4
t
(a) Piplining in modern CPU
(b) Parallel in modern GPU
...
(c) Selective view pipelining

Figure 1. Demonstration of the proposed selective view pipelining. Inspired by instruction pipelining in modern CPU (a), selective view pipelining (c) breaks the multi-view perception task into sequential steps, and only computes the chosen views (see Fig. 2) to minimize the computation. At the same time, computation across multiple objects (different border colors) can still enjoy parallelism in GPU (b). In this example, for a $N = 6$ view system, traditional computation (b) on GPU hardware that only supports two images at the same time will produce a throughput of 2 instances during 6 time steps. In contrast, by taking advantage of the parallelism, selective view pipelining computes 6 instances within 7 time steps.

In modern CPU, instruction pipelining [10] breaks individual instructions into steps to enable parallelism across multiple instructions within the same CPU (see Fig. 1a). For multi-view perception, although computations for different images can be paralleled over GPU [9], the level of parallelism is still limited by hardware. As an example, the GPU in Fig. 1b might only support parallel computing for two images at a time, resulting in at least 3 time steps for

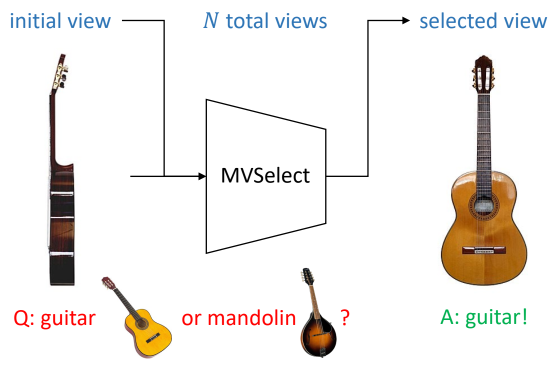


Figure 2. Efficient multi-view understanding with two glances. Instead of using all N views at once, a more efficient approach is to first examine one view and then select another view to resolve ambiguities from the initial glance. If the initial side view cannot distinguish between a guitar and a mandolin (a round-shaped instrument that may also have a flat back), we can then query the front view. This sequential process can work in a pipeline to enable parallelism between multiple

computing one object with 6 views.

In this work, we propose selective view pipelining, which breaks the multi-view perception task into a sequence of steps. Based on initial views, we estimate the most helpful views for this task, and only compute the selected views for maximum efficiency. The example in Fig. 1c shows that only selecting one additional view can greatly reduce the computation from 6 views to 2 views. By further taking advantage of the parallelism in GPUs, we can roughly improve the overall throughput by 3×.

To this end, this paper introduces a new angle to efficient multi-view understanding by selecting only the most useful views. To identify the best views, this approach leverages camera layouts, which is a key aspect overlooked by existing alternatives. With known camera layouts, networks should be able to infer what each camera view looks like and then choose accordingly, as previous works [20] have shown that networks can associate images with camera poses. Existing work on active vision [1, 8, 12] indicates that it is possible to find next-best-view (NBV) for reconstruction with 3D sensors. In this paper, we focus on recognition and detection problems using multiple RGB cameras, where a side view may confuse a guitar with a mandolin (Fig. 2). However, prior knowledge of the camera layout should inform the system that the front view can clear the ambiguities and should be queried.

To achieve this goal, this paper proposes a novel view selection module, MVSelect, that chooses the best camera views from any initial view at minimal computational costs. Using the same feature extracted for the final task of classification or detection, the proposed module analyzes the target object or scenario from existing views and then selects the

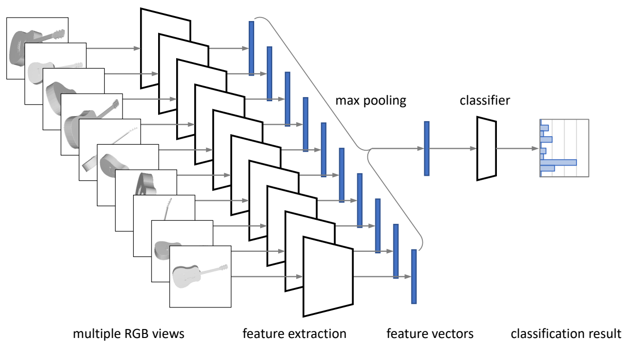


Figure 3. Multi-view classification with MVCNN [37].

next view that best helps the task (classification or detection) network. To navigate through the non-differentiable view selection (the system only looks at selected cameras, so not-selected views cannot back propagate gradients to the controller), following Mnih et al. [26], we train the controller via *trial-and-error*. This process is formulated as a reinforcement learning problem aiming to maximize the final recognition or detection accuracy.

On both multi-view classification and detection tasks, our experimental results show that MVSelect can provide a good strategy for fixed task networks, while also capable of joint training with the task network for further performance improvements. Specifically, when joint training both MVSelect and the task network, the resulting system can achieve competitive performance to using all N cameras, while using only 2 views for classification tasks and 3 views for detection tasks, respectively.

The computational overhead of MVSelect is very small, as it shares the feature extraction backbone with the task network and only has a few learnable layers. For the entire system, the computational cost is roughly proportional to the number of views used, *e.g.*, approximately $^2/_N$ of the total computation when 2 views are used, a significant efficiency boost.

The MVSelect policy also enables study on multi-view camera layout. In fact, we find that many of the N cameras are rarely chosen and can be shut off for further operational cost improvements, which can serve as a starting point for future study on multi-view camera layout optimization.

2. Background

Multi-view classification. One effective way for 3D shape recognition is to capture the object in multiple camera views. MVCNN [37] extracts feature vectors from the input views, and then uses max pooling to aggregate across multiple views for classification. Based on MVCNN, many alternative approaches are proposed. Qi et al. [30] propose sphere rendering at different volume resolutions. GVCNN [11] investigates hierarchical information between different

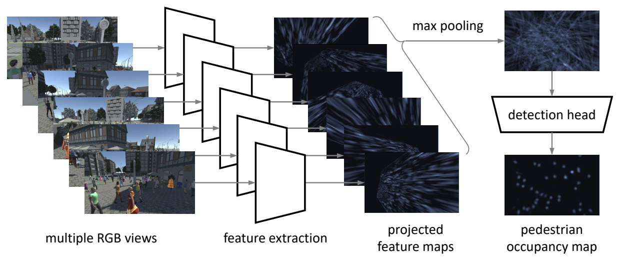


Figure 4. Multi-view detection with MVDet [17].

views by grouping the image features before the final aggregation. RotationNet [20] introduces a multi-task objective by jointly considering classification and camera poses. ViewGCN [42] uses a Graph Convolution Network (GCN) [23] instead of the max pooling layer to aggregate across views. Hamdi et al. [14] propose the MVTN network to estimate the best viewpoints for 3D point cloud models.

Multi-view detection estimate pedestrian occupancy from the bird's-eye-view (BEV). For this task, some methods [13, 33, 45] aggregate single-view detection results. Others find single-view detection results unreliable and instead aggregate the features. Hou et al. [17] introduce MVDet, the first fully deep-learning approach, which projects feature maps from each camera view to the BEV. Based on MVDet, researchers develop other deep methods. SHOT [36] projects the image feature map at different heights and stacks them together to improve performance. MVDeTr [16] deals with distinct distortion patterns from the projection. Qiu et al. [32] investigate data augmentation with simulated occlusion over multiple views.

Camera viewpoint study. Cao and Snavely [5] investigate this for reconstruction with structure-from-motion. Active vision community investigates next-best-view planning with *3D sensors* like RGBD cameras or LiDAR for scene reconstruction [4, 8, 35]. In multi-view classification, RotationNet [20] makes some pioneering work on limited view numbers by taking a random partial set of all N views of the image. MVTN [14] uses the 3D point cloud as initial input and then estimates the best camera layout for multi-view classification, but its moving camera assumption is hard to meet in real-world systems. In multi-view detection, Vora et al. [41] investigate camera layout generalization by randomly dropping camera views from both training and testing. Hou et al. [18] investigate how to place cameras. For 3D human pose estimation, Pirinen et al. [28] actively select cameras over a dome.

Reinforcement learning (RL) directs an agent to interact with the environment in a manner that maximizes cumulative rewards. State $s \in \mathcal{S}$, action $a \in \mathcal{A}$, and reward $r \in \mathcal{R}$ are key concepts to model the interaction between agent and environment. In a certain state s, policy $\pi(a|s)$ records the probability for each action, and state value function $V(s)$ estimates the future rewards when following the corresponding policy. To learn the best policy, Q-learning and DQN [25] optimize the action value function $Q(s,a)$, which describes the estimated future return for a specific action a at state s. Policy gradient methods like REINFORCE [43] and PPO [34] directly optimize for the polity $\pi(a|s)$.

3. Multi-view Network Revisit

3.1. Multi-view Classification with MVCNN

MVCNN [37] (Fig. 3) is a classic architecture which many multi-view classification networks build upon. Given N input images $\boldsymbol{x}_n, n \in \{1,\dots,N\}$, first, MVCNN uses its feature extractor $f(\cdot)$ to calculate the feature vectors,

$$\boldsymbol{h}_n = f(\boldsymbol{x}_n), \tag{1}$$

where the feature vector $\boldsymbol{h}_n \in \mathbb{R}^D$ is D-dimensional. Secondly, it uses max pooling to aggregate multiple views into an overall feature descriptor $\hat{\boldsymbol{h}} \in \mathbb{R}^D$,

$$\hat{\boldsymbol{h}} = \max_n \{\boldsymbol{h}_n\}, \tag{2}$$

where $\max\{\cdots\}$ takes the maximum along each of the D dimensions. Lastly, it applies the output head $g(\cdot)$ to produce the classification result $\hat{\boldsymbol{y}}$,

$$\hat{\boldsymbol{y}} = g\left(\hat{\boldsymbol{h}}\right). \tag{3}$$

In training, the original design by Su et al. [37] adopts a 2-stage paradigm by first training on individual views and then considering multiple views. In this paper, we skip the first stage and directly train MVCNN on all N views,

$$\mathcal{L}_{\text{MVCNN}} = \mathcal{L}_{\text{CE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}), \tag{4}$$

where $\mathcal{L}_{\text{CE}}(\cdot,\cdot)$ denotes the cross-entropy loss and $\boldsymbol{y}$ denotes the ground truth one-hot label.

3.2. Multi-view Detection with MVDet

MVDet [17] (Fig. 4) is a multi-view detection architecture that is followed by many recent works. To estimate human occupancy in bird's-eye-view (BEV), given input images $\boldsymbol{x}_n, n \in \{1,\dots,N\}$, MVDet first extracts D-channel feature maps for each view and uses perspective transformation to project the camera views to the BEV. Together, these operations can be considered as the BEV feature extraction step under Eq. 1, with the exception that $\boldsymbol{h}_n \in \mathbb{R}^{D\times H\times W}$ now denotes the D-channel feature map for the BEV scenario of shape $H \times W$. Secondly, for multi-view aggregation, instead of the concatenation in the original design, we choose element-wise max pooling in Eq. 2, producing the overall feature description $\hat{\boldsymbol{h}} \in \mathbb{R}^{D\times H\times W}$ that fits arbitrary numbers of views. Lastly, we apply the output head as

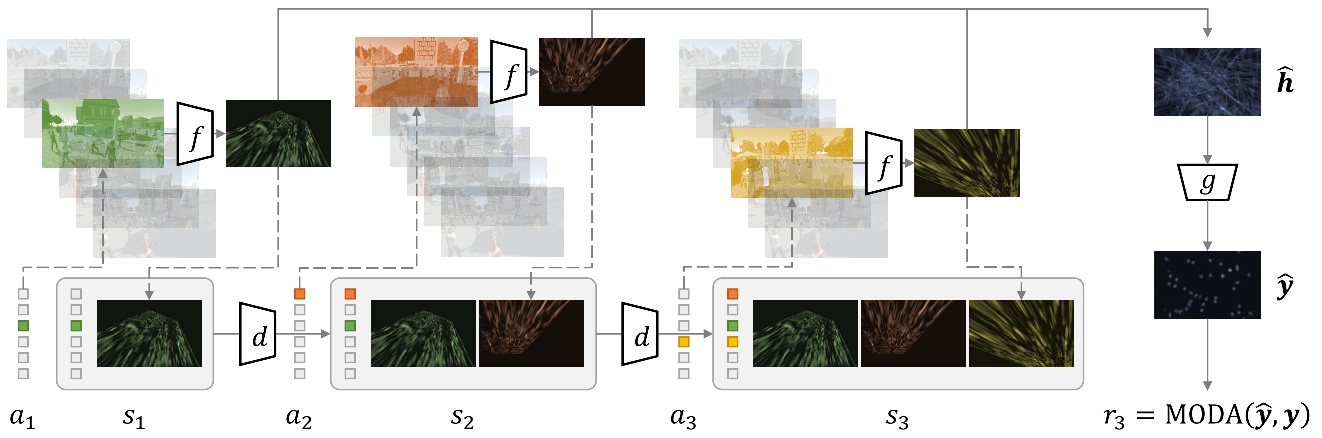


Figure 5. Efficient multi-view understanding with selective view pipelining using $T = 3$ glances. Solid lines indicate the network forward pass and dashed lines indicate the interaction between *agent* (MVSelect) and *environment* (multi-view system). The approach starts with a random initial view a_1, and the feature extractor $f(\cdot)$ computes its feature (Eq. 1). The *state* for this initial time step is recorded as s_1 (Eq. 6). Next, the proposed MVSelect module $d(\cdot)$ is used to choose a second view a_2, and the *state* is updated as s_2. Then, by repeating the last step, a third view a_3 is chosen and the state is updated as s_3. Finally, three views are aggregated into an overall descriptor $\hat{\boldsymbol{h}}$ (Eq. 2), and the task network output $\hat{\boldsymbol{y}}$ is calculated using the output head $g(\cdot)$ (Eq. 3). The final *reward* $r_3 = \text{MODA}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ is set as the accuracy fir the task network (Eq. 7), and all other rewards are set as zero.

Eq. 3 to generate a heatmap $\hat{\boldsymbol{y}} \in (0,1)^{H\times W}$ that indicates the likelihood of human occupancy in each BEV location. The loss for MVDet can be written as,

$$\mathcal{L}_{\text{MVDet}} = \mathcal{L}_{\text{BEV}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_n, \tag{5}$$

where $\mathcal{L}_{\text{BEV}}(\cdot,\cdot)$ denotes BEV output loss; $\boldsymbol{y} \in \{0,1\}^{H\times W}$ denotes binary ground truth map; and $\mathcal{L}_n$ denotes the auxiliary per-view loss with 2D bounding boxes.

4. Efficient Multi-view Understanding

In a multi-view system with N views, selective view planning uses a total of $T < N$ camera views $a_t \in \{1,\dots,N\}, t \in \{1,\dots,T\}$ for efficient understanding. To achieve this, we propose a view selection module, MVSelect, denoted as $d(\cdot)$, that sequentially selects camera views. Starting from a random initial view a_1, MVSelect chooses the remaining $T-1$ cameras by observing the target object or scene from existing views $a_1,\dots,a_t$ at each time step t and deciding which camera view a_{t+1} to select next for best performance. Once T camera views have been gathered, we aggregate them into an overall description $\hat{\boldsymbol{h}}$ using Eq. 2, and calculate the final output $\hat{\boldsymbol{y}}$ using Eq. 3. Fig. 5 gives an overview of the proposed efficient multi-view approach.

4.1. Problem Formulation

While iteratively selecting camera views, it is impossible to know what is in the *not-selected* camera views. If view selection was formulated as a prediction task in a supervised manner, the loss (final target metric at $t = T$) could not be back propagated to the not-selected views, making it *non-differentiable* and impossible to update the selection probability. Reinforcement learning, on the other hand, can update the controller even with the non-differentiable view selection. Therefore, we formulate this non-differentiable process as a reinforcement learning problem, where MVSelect is the *agent* and learns through *trial and error*, and the multi-view system is the *environment*.

State. In order to get a Markovian representation, we record the chosen camera views $a_1,\dots,a_t$ and the observations $\boldsymbol{h}_{a_1},\dots,\boldsymbol{h}_{a_t}$ as state s_t. We use the extracted features to represent the observations rather than the RGB camera views to reduce dimensionality and maximize efficiency, since these features will be used in the task network later (Eq. 2). Mathematically, we formulate the state s_t as,

$$\begin{aligned} s_t &= \left\langle s_t^{\text{cam}}, s_t^{\text{obs}} \right\rangle, \\ s_t^{\text{cam}} &= \sum_{\tau=1}^{t} \text{onehot}(a_\tau), \\ s_t^{\text{obs}} &= \max_{\tau=1}^{t} \{\boldsymbol{h}_{a_\tau}\}, \end{aligned} \tag{6}$$

where onehot $(\cdot)$ is the one-hot function over N cameras. This representation reflects both chosen cameras $s_t^{\text{cam}} \in \mathbb{R}^N$ and their observations $s_t^{\text{obs}} \in \mathbb{R}^D$, and maintains the same dimensionality across different time steps. The observation part s_t^{obs} also matches the overall representation in Eq. 2.

Action. For state $s_t, t \in \{1,\dots,T-1\}$, MVSelect takes the next camera view a_{t+1} as action.

Algorithm 1 Joint training of MVSelect and task network.

```
1: input: camera views $\boldsymbol{x}_n, n \in \{1, \dots, N\}$, ground truth $\boldsymbol{y}$, random initial view $a_1$, number of total views $T$, hyper-parameter $\epsilon$.
2: update: feature extractor $f(\cdot)$ and output head $g(\cdot)$ of task network, and MVSelect controller $d(\cdot)$.
3: initialize the total loss $\mathcal{L}_{\text{total}} = 0$;
4: for $t \in \{1, \dots, T-1\}$ do
5:     select and apply the next action using $\epsilon$-greedy: with probability $\epsilon$ adopt a random action, or else choose the action with highest value $a_{t+1} = \arg\max_a Q(s_t, a)$;
6:     observe next state $s_{t+1}$ (Eq. 6) and reward $r_{t+1}$ (Eq. 7);
7:     calculate the RL loss $\mathcal{L}_{\text{RL}}$ (Eq. 9) and update the total loss $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{total}} + \mathcal{L}_{\text{RL}}$;
8: end for
9: calculate the task loss $\mathcal{L}_{\text{task}}$ (Eq. 4 or Eq. 5) and update the total loss $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{total}} + \mathcal{L}_{\text{task}}$;
10: optimize for the total loss $\mathcal{L}_{\text{total}}$.
```

Reward. Upon taking action a_{t+1}, the system receives reward r_{t+1} and transitions into the next state s_{t+1}. To achieve high task network performance, we consider the following as reward,

$$\begin{aligned} r_t &= 0, t \in \{1, \dots, T-1\}, \\ r_T^{\text{MVCNN}} &= \mathbb{1}(\hat{\boldsymbol{y}} = \boldsymbol{y}), \; r_T^{\text{MVDet}} = \text{MODA}(\hat{\boldsymbol{y}}, \boldsymbol{y}), \end{aligned} \tag{7}$$

where $\mathbb{1}(\cdot)$ denotes the binary indicator function, $\text{MODA}(\cdot,\cdot)$ is the evaluation metric for multi-view detection [21]. We also experiment with other reward designs in Section 5.4.

4.2. MVSelect Architecture

We design MVSelect architecture $d(\cdot)$ with two branches. The first branch expands the camera selection result $s_t^{\text{cam}} \in \mathbb{R}^N$ into D-dimensional learnable camera embeddings, and then sums over the selected embeddings to formulate a hidden vector. The second branch converts the observation $s_t^{\text{obs}} \in \mathbb{R}^D$ into another hidden vector. By combining the two hidden vectors, the controller network outputs the action-value $Q(s, a)$, which measures the expected cumulative rewards for taking an action a in a given state s. See Supplementary Materials for figure illustrations.

During testing, MVSelect outputs the next action as,

$$a_{t+1} = \arg\max_a Q(s_t, a),$$

which maximizes the expected cumulative rewards.

4.3. Training Scheme

We adopt Q-learning [38] for training MVSelect. Specifically, action-value function $Q(\cdot,\cdot)$ should estimate the cumulative future rewards after taking action a_{t+1} at state s_t,

$$Q(s_t, a_{t+1}) = \mathbb{E}\left(\sum_{\tau=t+1}^{T} \gamma^{\tau-t-1} r_\tau\right),$$

where $\mathbb{E}(\cdot)$ denotes the expectation, and $\gamma \in [0, 1]$ denotes the discount factor. We take the temporal difference (TD) [38] target as supervision for the action value,

$$q_t = \begin{cases} r_{t+1} + \gamma \max_a Q(s_{t+1}, a), & \text{if } t < T-1 \\ r_T, & \text{otherwise} \end{cases}, \tag{8}$$

and calculate the loss using the L_2 distance,

$$\mathcal{L}_{\text{RL}} = \sum_{t=1}^{T-1} \mathcal{L}_{\text{MSE}}(Q(s_t, a_{t+1}), q_t), \tag{9}$$

where the next action a_{t+1} is chosen using ϵ-greedy for exploration-exploitation trade offs.

In joint training, the task network takes supervision from $\mathcal{L}_{\text{task}}$ (Eq. 4 and Eq. 5), and the selection module takes supervision from $\mathcal{L}_{\text{RL}}$ (Eq. 9). A step-by-step demonstration of this process can be found in Algorithm 1.

5. Experiments

5.1. Experiment Settings

Datasets. For multi-view classification, we use synthetic dataset ModelNet40 [44] under two different camera layouts (12 views and 20 views) and real-world dataset ScanObjectNN (12 views) [39]. For multi-view detection, we use real-world dataset Wildtrack (7 views) [6] and synthetic dataset MultiviewX (6 views) [17].

Evaluation metrics. For multi-view classification, we report instance-averaged accuracy. Regarding multi-view detection, we report multi-object detection accuracy (MODA), which is calculated as $1 - \frac{\text{FP+FN}}{\text{GT}}$ [21]. All metrics are reported in percentages.

Implementation details. For multi-view classification, we input images of size 224×224 to the MVCNN model. For multi-view detection, we use a resolution of 720×1280 with view-coherent data augmentation [16], and downsample the BEV grid by a factor of 4. In terms of architecture, we use ResNet-18 [15] as feature extractor $f(\cdot)$.

We train all networks for 10 epochs using the Adam optimizer [22]. We use learning rates of 5×10^{-5} and 5×10^{-4}, with batch sizes of 8 and 1 for MVCNN and MVDet, respectively. The MVSelect module is trained using a learning rate of 1×10^{-4}. For joint training, we decrease the learning rate for the task network to 1/5 of its original value.

Table 1. Performance comparison with state-of-the-art methods. Results are averaged from 5 runs. * indicates that the camera poses are dynamically chosen and do not follow a pre-defined layout. † indicates our implementation.

	view selection ($T=2$)	ModelNet40		ScanObjNN
		12 views	20 views	
MVCNN [37]	N/A	90.1	92.0	-
GVCNN [11]	N/A	92.6	-	-
MHBN [47]	N/A	93.4	-	-
RotationNet [20]	N/A	-	94.7	-
RelationNet [46]	N/A	94.3	97.3	-
ViewGCN [42]	N/A	-	**97.6**	-
MVCNN†	N/A	**94.5**	96.5	86.1
PointNet [31] + ViewGCN	MVTN* ($T=12$ or 20) [14]	93.8	93.5	**92.6**
MVCNN†	SEE++† [3]	79.2	70.3	75.5
MVCNN†	Potthast and Sukhatme† [29]	76.5	66.2	72.1
MVCNN†	MVSelect	88.2	79.6	80.0
MVCNN† (joint training)	MVSelect	94.3	94.4	84.1

	view selection ($T=3$)	Wildtrack	MultiviewX
RCNN & cluster [45]	N/A	11.3	18.7
POM-CNN [13]	N/A	23.2	-
DeepMCD [7]	N/A	67.8	70
Deep-Occlusion [2]	N/A	74.1	75.2
MVDet [17]	N/A	88.2	83.9
SHOT [36]	N/A	90.2	88.3
MVDeTr [16]	N/A	**91.5**	**93.7**
MVDet†	N/A	90.0	93.0
MVDet†	SEE++† [3]	76.1	77.8
MVDet†	Potthast and Sukhatme† [29]	78.3	78.0
MVDet†	max FoV† [35]	78.0	73.9
MVDet†	MVSelect	80.0	78.7
MVDet† (joint training)	MVSelect	88.6	88.1

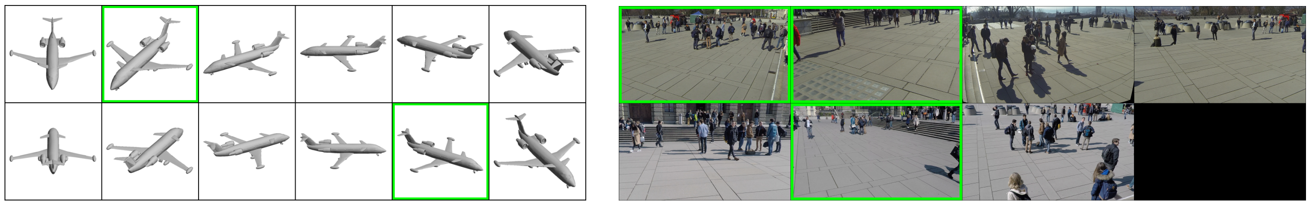


Figure 6. Example of selected views on ModelNet40 (left) and Wildtrack (right).

Regarding hyperparameters, we set the future reward discount factor $\gamma = 0.99$, and the exploration ratio ϵ to gradually decrease from 0.95 to 0.05 during training. All experiments are conducted on a single RTX-3090 GPU and averaged across 5 repetitive runs.

Experimental setups. First, we fix the task network. We compare with state-of-the-art methods including 1) NBV policy with depth-sensing enabled point-based occlusion estimation [3], 2) NBV policy with depth-sensing enabled voxel-based probabilistic representation [29], and 3) NBV policy to maximize field-of-view (FoV) coverage [35] (only applicable to multi-view detection). We also report two baselines: 1) random selection, 2) the best policy on the validation set; and two oracles: for a certain initial view, 1) choosing the overall best-performing camera for all instances in the dataset (*dataset-level oracle*) and 2) choosing specifically for each instance (*instance-level oracle*). For the two oracles, the former reflects the upper bound of performance achievable by human-designed heuristics, and the latter represents the theoretical performance ceiling when keeping the task network fixed.

Second, we jointly train the proposed view selection module with the task networks. For this experiment, our goal is to achieve the highest possible performance using T views. If not specified, we use a total of $T = 2$ views for multi-view classification and $T = 3$ views for detection.

5.2. Evaluation of MVSelect

For **multi-view classification**, as in Table 1, MVSelect with fixed multi-view classification network achieves competitive performance compared to other state-of-the-art view selection methods. When compared with baselines and oracles in Table 2, the proposed view selection module, can choose the views effectively. Compared to dataset-level oracles (same policy for all instances with the same initial camera), MVSelect also turns out to be advantageous by 3.0%, 9.7%, and 1.5% across the two settings. This verifies that MVSelect can take the target object into consideration (see Fig. 2) and select different cameras for different instances under the same initial camera.

When joint training MVCNN with MVSelect, we witness large improvements. In fact, on two settings, the results are only 0.2%, 2.1%, and 2.0% behind compared to the full N-view system. Overall, we believe that MVSelect and its joint training capabilities enable us to consider only 2 views without major performance drawbacks. We demonstrate an example of the selected views in Fig. 6, and the MVSelect policy in Fig. 7.

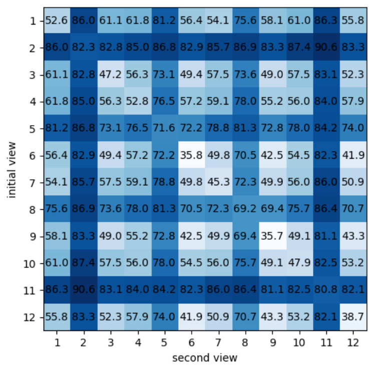
For **multi-view detection**, MVSelect achieves competitive performance compared to existing NBV methods, some of them even require additional depth sensing (Table 1). Although the raw improvements are not as substantial as those in multi-view classification, they are **statistically highly significant** (p-value < 0.001). This shows human designed heuristics like minimal occlusion or maximum FoV coverage cannot guarantee the best detection results.

When compared to baselines and oracles, MVSelect also shows to be competitive (see Table 2). Specifically, we observe that the instance-level oracle remains relatively low compared to that of multi-view classification tasks. This is

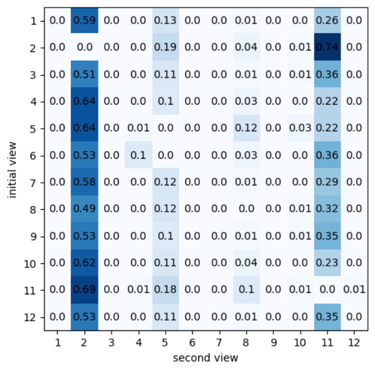
Table 2. Comparison with view selection baselines and oracles on multi-view classification and detection datasets.

view selection ($T = 2$)	ModelNet40		ScanObjNN
	12 views	20 views	
N/A: all N views	94.5	96.5	86.1
dataset-lvl oracle	85.2 ± 1.4	69.9 ± 1.9	78.5 ± 1.2
instance-lvl oracle	96.5 ± 0.4	98.1 ± 1.9	93.4 ± 0.6
random selection	71.5 ± 2.5	48.1 ± 3.1	74.2 ± 3.5
validation best policy	85.1 ± 0.8	69.5 ± 1.0	77.5 ± 1.1
MVSelect	88.2 ± 0.4	79.6 ± 1.8	80.0 ± 0.8
MVSelect + joint training	94.3 ± 0.2	94.4 ± 0.2	84.1 ± 0.2

view selection ($T = 3$)	Wildtrack	MultiviewX
N/A: all N views	90.0	93.0
dataset-lvl oracle	82.5 ± 0.4	80.2 ± 0.3
instance-lvl oracle	87.4 ± 0.4	82.3 ± 0.8
random selection	74.9 ± 1.3	76.2 ± 1.4
validation best policy	79.5 ± 1.1	78.0 ± 0.4
MVSelect	80.0 ± 0.8	78.7 ± 0.5
MVSelect + joint training	88.6 ± 0.2	88.1 ± 0.2



(a) Classification accuracy



(b) MVSelect policy

Figure 7. Multi-view classification with $T = 2$ views on the 12-view setup of ModelNet40. The task network is fixed once trained. **Left**: test set accuracy of using two views. **Right**: MVSelect policy for the test set.

Table 3. Computation efficiency.

	views	FLOPs			throughput (instance/s)
		$f(\cdot)$	$g(\cdot)$	$d(\cdot)$	
ModelNet40	20	36.5G	20.5k	N/A	119.6
	2	3.6G	20.5k	1.1M	361.8
	12	21.9G	20.5k	N/A	196.2
	2	3.6G	20.5k	530.4k	507.9
Wildtrack	7	1.2T	19.1G	N/A	8.4
	3	511.6G	19.1G	3.2G	16.2
MultiviewX	6	1.0T	17.7G	N/A	9.8
	3	511.6G	17.7G	2.9G	16.9

Table 4. Ablation study.

	ModelNet40	Wildtrack
	12 views	
MVSelect	88.2	80.0
w/o camera branch	88.2	79.1
w/o feature branch	85.0	79.7
w/ transformer	87.0	78.7
reward=Δ task loss	88.0	80.1

likely due to the target scenarios not being fully captured by any single view, and the multi-view detection network needs multiple views to collaborate with each other for optimal results. Compared to the dataset-level oracles, we find the MVSelect policy lose its edge. In fact, we find that for multi-view detection, MVSelect tends to select the same camera for a given initial view, since it is *not* aware of the situation outside of the FoV coverage. As a result, it cannot choose cameras based on *uncovered* areas in different frames. Without this instance-aware advantage, the fixed camera policy learned during training (MVSelect) cannot outperform the dataset-level oracle, whose policy is computed on the test set.

Joint training with MVDet once again leads to substantial performance improvements over keeping the task network fixed. In fact, the results even exceed the instance-level oracle for fixed task networks. Using $T = 3$ views, the joint training approach provides competitive results to using all N views, and exceeds the reported performance in the original MVDet paper [17]. We present an example of the selected views in the Wildtrack dataset in Fig. 6.

5.3. Efficiency Analysis

In Table 3, following previous works in efficient inference [19, 24], we detail the computational cost in FLOPs for task networks and MVSelect. Specifically, we find feature extraction $f(\cdot)$ to take up the majority of the computation, while everything else is lighter by at least an order of magnitude. Overall, we verify that using $T = 2$ or 3 out of N views can reduce the computational cost to roughly T/N.

In terms of inference speed, we find reduction in FLOPs results in monotonically increasing throughputs, ranging from 1.72× to 3.03×. Due to factors such as implementation, parallelization, and hardware limitations, actual speedups cannot actually reach the level of computational cost reduction, as suggested by previous study [27].

5.4. Variant Study

Ablation study. Regarding the MVSelect architecture design, Table 4 shows that removing the camera branch and feature branch primarily affects multi-view detection and multi-view classification performance, respectively. This aligns with the policy we learned for the two tasks. For multi-view detection, since the system has no clue about ar-

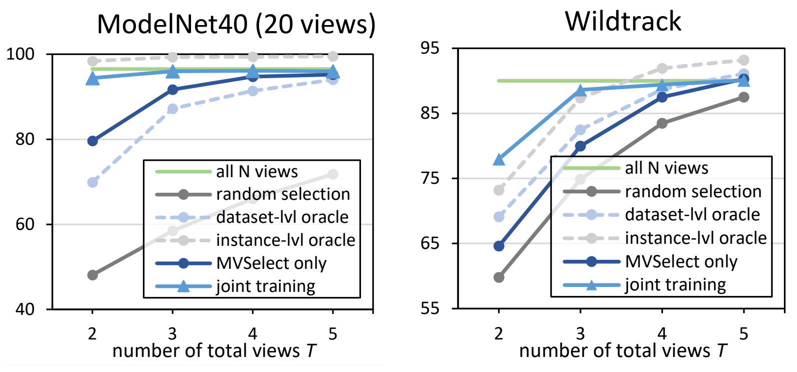


Figure 9. System performance under different number of total views T. We report classification accuracy and MODA for the two tasks, respectively (same for Fig. 10 and Table. 4). Circle and triangle markers indicate whether the task network is fixed or not, respectively. Dotted lines represent oracle performance (not achievable).

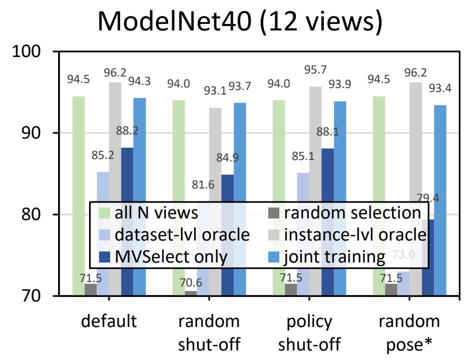


Figure 10. System evaluation under different settings. Except for the random pose* setting (where MVSelect variants are re-trained), all models are trained on the default setting.

eas outside camera FoVs, the camera branch plays a more important role for encoding prior knowledge of the scene layout. For multi-view classification, however, target objects are fully observable, and MVSelect can make different decisions for each instance. Thus, the feature branch is more important, as it enables per-instance decision making.

When changing the MVSelect network architecture into transformer [40], we find that more parameters do not translate into better performance.

Another option for the reward is the change in task loss. In our variant study, this reward design does not show any significant advantages over the current design.

Influence of total view count. The performance curves in Fig. 9 demonstrate that the joint training variant achieves competitive performance with as few as $T = 2$ views for multi-view classification and $T = 3$ views for multi-view detection, beyond which point performance plateaus. By contrast, learning MVSelect for fixed task networks shows a less steep curve. The performance increases up to a total of $T = 5$ views, at which point the MVSelect policy performs comparably to the full N-view system.

Camera layout optimization. Determining the optimal camera locations is crucial for setting up an effective multi-view system. In Fig.7, we observe that not all cameras are equally useful according to the MVSelect policy. To address this, we allocate a validation partition of the data to identify the more useful camera views and then disable half of the $N = 12$ cameras that are not frequently utilized. In testing, we find that some cameras are not as useful ("policy shut-off" in Fig. 10) and the proposed method can generalize to test-time disturbances including malfunctioning cameras ("random shut-off" in Fig.10). Although it is necessary to set up all N cameras for the analysis, optimizing the multi-view camera layout can be a crucial step towards achieving optimal performance, and merits further investigation.

Random object pose. In real-world applications of multi-view systems, such as those found in iPhones and Teslas, the cameras may remain fixed in their relative positions while the entire system is in motion. To simulate this scenario in our experiments, we introduce the random object pose setting (Fig. 10) and re-train MVSelect. While there is no exact object pose as supervision, the reinforcement learning approach is able to roughly infer the relative poses between the object and the multi-view system, resulting in improved performance compared to random selection and dataset-level oracle (which is arguably inappropriate for this setup). In the future, we plan to estimate camera poses with respect to the object or the environment as an auxiliary supervision for moving setups.

6. Conclusion

In conclusion, this paper proposes selective view pipelining, an efficient approach for multi-view understanding by limiting the number of views. To this end, a camera view selection module, MVSelect, is proposed along with a reinforcement learning based training scheme that can learn from the non-differentiable selection process. When jointly trained with the task network, the proposed approach demonstrates competitive performance on multi-view classification and detection tasks at fractions of the computational cost. Overall, the proposed efficient approach provides an alternative to reducing image resolution and using lighter networks, and paves ways for future multi-view camera layout optimization.

Acknowledgement

This work was supported by the ARC Discovery Project (DP210102801).

References

[1] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International journal of computer vision*, 1: 333–356, 1988. 2

[2] Pierre Baqué, François Fleuret, and Pascal Fua. Deep occlusion reasoning for multi-camera multi-target detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 271–279, 2017. 6

[3] Rowan Border and Jonathan D Gammell. Proactive estimation of occlusions and scene coverage for planning next best views in an unstructured representation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4219–4226. IEEE, 2020. 6

[4] Rowan Border, Jonathan D Gammell, and Paul Newman. Surface edge explorer (see): Planning next best views directly from 3d observations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6116–6123. IEEE, 2018. 3

[5] Song Cao and Noah Snavely. Minimal scene descriptions from structure from motion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 461–468, 2014. 3

[6] Tatjana Chavdarova, Pierre Baqué, Stéphane Bouquet, Andrii Maksai, Cijo Jose, Timur Bagautdinov, Louis Lettry, Pascal Fua, Luc Van Gool, and François Fleuret. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5030–5039, 2018. 1, 5

[7] Tatjana Chavdarova et al. Deep multi-camera people detection. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 848–853. IEEE, 2017. 6

[8] Shengyong Chen, Youfu Li, and Ngai Ming Kwok. Active vision in robotic systems: A survey of recent developments. *The International Journal of Robotics Research*, 30 (11):1343–1377, 2011. 2, 3

[9] Wikipedia Contributors. Graphics processing unit, 2019. 1

[10] Wikipedia Contributors. Instruction pipelining, 2020. 1

[11] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 264–272, 2018. 2, 6

[12] John M Findlay and Iain D Gilchrist. *Active vision: The psychology of looking and seeing*. Number 37. Oxford University Press, 2003. 2

[13] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):267–282, 2007. 3, 6

[14] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1–11, 2021. 3, 6

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[16] Yunzhong Hou and Liang Zheng. Multiview detection with shadow transformer (and view-coherent data augmentation). In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1673–1682, 2021. 3, 5, 6

[17] Yunzhong Hou, Liang Zheng, and Stephen Gould. Multiview detection with feature perspective transformation. In *European Conference on Computer Vision*, pages 1–18. Springer, 2020. 1, 3, 5, 6, 7

[18] Yunzhong Hou, Xingjian Leng, Tom Gedeon, and Liang Zheng. Optimizing camera configurations for multi-view pedestrian detection. *arXiv preprint arXiv:2312.02144*, 2023. 3

[19] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 7

[20] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5010–5019, 2018. 2, 3, 6

[21] Rangachar Kasturi, Dmitry Goldgof, Padmanabhan Soundararajan, Vasant Manohar, John Garofolo, Rachel Bowers, Matthew Boonstra, Valentina Korzhova, and Jing Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):319–336, 2008. 5

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. 5

[23] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*, 2016. 3

[24] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017. 7

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 3

[26] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014. 2

[27] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 1, 7

[28] Aleksis Pirinen, Erik Gärtner, and Cristian Sminchisescu. Domes to drones: Self-supervised active triangulation for 3d human pose reconstruction. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[29] Christian Potthast and Gaurav S Sukhatme. A probabilistic framework for next best view estimation in a cluttered

environment. *Journal of Visual Communication and Image Representation*, 25(1):148–164, 2014. 6

[30] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 1, 2

[31] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 6

[32] Rui Qiu, Ming Xu, Yuyao Yan, Jeremy S Smith, and Xi Yang. 3d random occlusion and multi-layer projection for deep multi-camera pedestrian localization. *arXiv preprint arXiv:2207.10895*, 2022. 3

[33] Gemma Roig, Xavier Boix, Horesh Ben Shitrit, and Pascal Fua. Conditional random fields for multi-camera object detection. In *2011 International Conference on Computer Vision*, pages 563–570. IEEE, 2011. 3

[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 3

[35] William R Scott, Gerhard Roth, and Jean-François Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys (CSUR)*, 35(1):64–96, 2003. 3, 6

[36] Liangchen Song, Jialian Wu, Ming Yang, Qian Zhang, Yuan Li, and Junsong Yuan. Stacked homography transformations for multi-view pedestrian detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6049–6057, 2021. 3, 6

[37] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 1, 2, 3, 6

[38] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 5

[39] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019. 5

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 8

[41] Jeet Vora, Swetanjal Dutta, Kanishk Jain, Shyamgopal Karthik, and Vineet Gandhi. Bringing generalization to deep multi-view detection. *arXiv preprint arXiv:2109.12227*, 2021. 3

[42] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1850–1859, 2020. 3, 6

[43] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992. 3

[44] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 5

[45] Yuanlu Xu, Xiaobai Liu, Yang Liu, and Song-Chun Zhu. Multi-view people tracking via hierarchical trajectory composition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4256–4265, 2016. 3, 6

[46] Ze Yang and Liwei Wang. Learning relationships for multi-view 3d object recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7505–7514, 2019. 6

[47] Tan Yu, Jingjing Meng, and Junsong Yuan. Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 186–194, 2018. 6