RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints

Asako Kanezaki$^1$, Yasuyuki Matsushita$^2$, and Yoshifumi Nishida$^1$

$^1$National Institute of Advanced Industrial Science and Technology (AIST)
$^2$Graduate School of Information Science and Technology, Osaka University

Abstract

We propose a Convolutional Neural Network (CNN)-based model “RotationNet,” which takes multi-view images of an object as input and jointly estimates its pose and object category. Unlike previous approaches that use known viewpoint labels for training, our method treats the viewpoint labels as latent variables, which are learned in an unsupervised manner during the training using an unaligned object dataset. RotationNet is designed to use only a partial set of multi-view images for inference, and this property makes it useful in practical scenarios where only partial views are available. Moreover, our pose alignment strategy enables one to obtain view-specific feature representations shared across classes, which is important to maintain high accuracy in both object categorization and pose estimation. Effectiveness of RotationNet is demonstrated by its superior performance to the state-of-the-art methods of 3D object classification on 10- and 40-class ModelNet datasets. We also show that RotationNet, even trained without known poses, achieves the state-of-the-art performance on an object pose estimation dataset.

1. Introduction

Object classification accuracy can be enhanced by the use of multiple different views of a target object [4, 23]. Recent remarkable advances in image recognition and collection of 3D object models enabled the learning of multi-view representations of objects in various categories. However, in real-world scenarios, objects can often only be observed from limited viewpoints due to occlusions, which makes it difficult to rely on multi-view representations that are learned with the whole circumference. The desired property for the real-world object classification is that, when a viewer observes a partial set (≥ 1 images) of the full multi-view images of an object, it should recognize from which directions it observed the target object to correctly infer the category of the object. It has been understood that if the viewpoint is known the object classification accuracy can be improved. Likewise, if the object category is known, that helps infer the viewpoint. As such, object classification and viewpoint estimation is a tightly coupled problem, which can best benefit from their joint estimation.

We propose a new Convolutional Neural Network (CNN) model that we call RotationNet, which takes multi-view images of an object as input and predicts its pose and object category (Fig. 1). RotationNet outputs viewpoint-specific category likelihoods corresponding to all pre-defined discrete viewpoints for each image input, and then selects the object pose that maximizes the integrated object category likelihood. Whereas, at the training phase, RotationNet uses a complete set of multi-view images of an object captured from all the pre-defined viewpoints, for inference it is able to work with only a partial set of all the multi-view images – a single image at minimum – as input. In addition, RotationNet does not require the multi-view images to be provided at once but allows their sequential input.
and updates of the target object’s category likelihood. This property is suitable for applications that require on-the-fly classification with a moving camera.

The most representative feature of RotationNet is that it treats viewpoints where training images are observed as latent variables during the training (Fig. 2). This enables unsupervised learning of object poses using an unaligned object dataset; thus, it eliminates the need of preprocessing for pose normalization that is often sensitive to noise and individual differences in shape. Our method automatically determines the basis axes of objects based on their appearance during the training and achieves not only intra-class but also inter-class object pose alignment. Inter-class pose alignment is important to deal with joint learning of object pose and category, because the importance of object classification lies in emphasizing differences in different categories when their appearances are similar. Without inter-class pose alignment, it may become an ill-posed problem to obtain a model to distinguish, e.g., a car and a bus if the side view of a car is compared with the frontal view of a bus.

Our main contributions are described as follows. We first show that RotationNet outperforms the current state-of-the-art classification performance on 3D object benchmark datasets consisting of 10- and 40-categories by a large margin (Table 5). Next, even though it is trained without the ground-truth poses, RotationNet achieves superior performance to previous works on an object pose estimation dataset. We also show that our model generalizes well to a real-world image dataset that was newly created for the general task of multi-view object classification. Finally, we train RotationNet with the new dataset named MIRO and demonstrate the performance of real-world applications using a moving USB camera or a head-mounted camera (Microsoft HoloLens) in our supplementary video.

2. Related work

There are two main approaches for the CNN-based 3D object classification: voxel-based and 2D image-based approaches. The earliest work on the former approach is 3D ShapeNets [39], which learns a Convolutional Deep Belief Network that outputs probability distributions of binary occupancy voxel values. Latest works on similar approaches showcased improved performance [21][20][38]. Even when working with 3D objects, 2D image-based approaches are shown effective for general object recognition tasks. Su et al. [34] proposed multi-view CNN (MVCNN), which takes multi-view images of an object captured from surrounding virtual cameras as input and outputs the object’s category label. Multi-view representations are also used for 3D shape retrieval [1]. Qi et al. [25] gives a comprehensive study on the voxel-based CNNs and multi-view CNNs for 3D object classification. Other than those above, point-based approach [11][24][15] is recently drawing much attention; however, the performance on 3D object classification is yet inferior to those of multi-view approaches. The current state-of-the-art result on the ModelNet40 benchmark dataset is reported by Wang et al. [37], which is also based on the multi-view approach.

Because MVCNN integrates multi-views in a view-pooling layer which lies in the middle of the CNN, it requires a complete set of multi-view images recorded from all the pre-defined viewpoints for object inference. Unlike MVCNN, our method is able to classify an object using a partial set of multi-view images that may be sequentially observed by a moving camera. Elhoseiny et al. [9] explored CNN architectures for joint object classification and pose estimation learned with multi-view images. Whereas their method takes a single image as input for its prediction, we mainly focus on how to aggregate predictions from multiple images captured from different viewpoints.

Viewpoint estimation is significant in its role in improving object classification. Better performance was achieved on face identification [45], human action classification [7], and image retrieval [36] by generating unseen views after observing a single view. These methods “imagine” the appearance of objects’ unobserved profiles, which is inherently more uncertain than using real observations. Sedaghat et al. [29] proposed a voxel-based CNN that outputs orientation labels as well as classification labels and demonstrated that it improved 3D object classification performance.

All the methods mentioned above assume known poses in training samples; however, object poses are not always aligned in existing object databases. Novotny et al. [22] proposed a viewpoint factorization network that utilizes relative pose changes within each sequence to align objects in videos in an unsupervised manner. Our method also aligns object poses via unsupervised viewpoint estimation, where viewpoints of images are treated as latent variables during the training. Here, viewpoint estimation is learned in an unsupervised manner to best promote the object categorization task. In such a perspective, our method is related to Zhou et al. [44], where viewpoint synthesis is trained as the “meta”-task to train multi-view pose networks by utilizing the synthesized views as the supervisory signal.

Although joint learning of object classification and pose estimation has been widely studied [28][19][42][2][35], inter-class pose alignment has drawn little attention. However, it is beneficial to share view-specific appearance information across classes to simultaneously solve for object classification and pose estimation. Kuznetsova et al. [17] pointed out this issue and presented a metric learning approach that shares visual components across categories for simultaneous pose estimation and class prediction. Our method also uses a model with view-specific appearances that are shared across classes; thus, it is able to maintain high accuracy for both object classification and pose estimation.
Figure 2. Illustration of the training process of RotationNet, where the number of views $M$ is 3 and the number of categories $N$ is 2. A training sample consists of $M$ images of an unaligned object and its category label $y$. For each input image, our CNN (RotationNet) outputs $M$ histograms with $N + 1$ bins whose norm is 1. The last bin of each histogram represents the “incorrect view” class, which serves as a weight of how likely the histogram does not correspond to each viewpoint variable. According to the histogram values, we decide which viewpoint variable is incorrect, i.e., view $i$ is incorrect, $P(\hat{y}_i = y | x_i, v_i)$ may not necessarily be high because the image $x_i$ is captured from a different viewpoint. As described above, we decide the viewpoint variable $v_i$ according to the $P(\hat{y}_i = y | x_i, v_i)$ outputs as in (1). In order to obtain a stable solution of $\{v_i\}_{i=1}^M$ in (1), we introduce an “incorrect view” class and append it to the target category classes. Here, the “incorrect view” class plays a similar role to the “background” class for object detection tasks, which represents negative samples that belong to a “non-target” class. Then, RotationNet calculates $P(\hat{y}_i | x_i, v_i)$ by applying softmax functions to the $(N+1)$-dimensional outputs, where $\sum_{j=1}^{N+1} P(\hat{y}_i | x_i, v_i) = 1$. Note that $P(\hat{y}_i = N + 1 | x_i, v_i)$, which corresponds to the probability that the image $x_i$ belongs to the “incorrect view” class for the $v_i$-th viewpoint, indicates how likely it is that the estimated viewpoint variable $v_i$ is incorrect.

Based on the above discussion, we substantiate (1) as follows. Letting $P_i = \{p_{j,k}^{(i)}\} \in \mathbb{R}_+^{M \times (N+1)}$ denote a matrix composed of $P(\hat{y}_i | x_i, v_j)$ for all the $M$ viewpoints and $N + 1$ classes, the target value of $P_i$ in the case that $v_i$ is correctly estimated is defined as follows:

$$p_{j,k}^{(i)} = \begin{cases} 1 & (j = v_i \text{ and } k = y) \text{ or } (j \neq v_i \text{ and } k = N + 1) \\ 0 & \text{(otherwise)} \end{cases}$$

In this way, (1) can be rewritten as the following cross-entropy optimization problem:

$$\max_{R, \{v_i\}_{i=1}^M} \sum_{i=1}^M \left( \log p_{v_i,y}^{(i)} + \sum_{j \neq v_i} \log p_{j,N+1}^{(i)} \right).$$

3. Proposed method

The training process of RotationNet is illustrated in Fig. 2. We assume that multi-view images of each training object instance are observed from all the pre-defined viewpoints. Let $M$ be the number of the pre-defined viewpoints and $N$ denote the number of target object categories. A training sample consists of $M$ images of an object $\{x_i\}_{i=1}^M$ and its category label $y \in \{1, \ldots, N\}$. We attach a viewpoint variable $v_i \in \{1, \ldots, M\}$ to each image $x_i$ and set it to $j$ when the image is observed from the $j$-th viewpoint, i.e., $v_i \leftarrow j$. In our method, only the category label $y$ is given during the training whereas the viewpoint variables $\{v_i\}$ are unknown, namely, $\{v_i\}$ are treated as latent variables that are optimized in the training process.

RotationNet is defined as a differentiable multi-layer neural network $R(\cdot)$. The final layer of RotationNet is the concatenation of $M$ softmax layers, each of which outputs the category likelihood $P(\hat{y}_i | x_i, v_i = j)$ where $j \in \{1, \ldots, M\}$ for each image $x_i$. Here, $\hat{y}_i$ denotes an estimate of the object category label for $x_i$. For the training of RotationNet, we input the set of images $\{x_i\}_{i=1}^M$ simultaneously and solve the following optimization problem:

$$\max_{R, \{v_i\}_{i=1}^M} \prod_{i=1}^M P(\hat{y}_i = y | x_i, v_i).$$

The parameters of $R$ and latent variables $\{v_i\}_{i=1}^M$ are optimized to output the highest probability of $y$ for the input of multi-view images $\{x_i\}_{i=1}^M$.

Now, we describe how we design $P(\hat{y}_i | x_i, v_i)$ outputs. First of all, the category likelihood $P(\hat{y}_i = y | x_i, v_i)$ should become close to one when the estimated $v_i$ is correct; in other words, the image $x_i$ is truly captured from the $v_i$-th viewpoint. Otherwise, in the case that the estimated $v_i$ is incorrect, $P(\hat{y}_i = y | x_i, v_i)$ may not necessarily be high because the image $x_i$ is captured from a different viewpoint. As described above, we decide the viewpoint variable $v_i$ according to the $P(\hat{y}_i = y | x_i, v_i)$ outputs as in (1). In order to obtain a stable solution of $\{v_i\}_{i=1}^M$ in (1), we introduce an “incorrect view” class and append it to the target category classes. Here, the “incorrect view” class plays a similar role to the “background” class for object detection tasks, which represents negative samples that belong to a “non-target” class. Then, RotationNet calculates $P(\hat{y}_i | x_i, v_i)$ by applying softmax functions to the $(N+1)$-dimensional outputs, where $\sum_{j=1}^{N+1} P(\hat{y}_i | x_i, v_i) = 1$. Note that $P(\hat{y}_i = N + 1 | x_i, v_i)$, which corresponds to the probability that the image $x_i$ belongs to the “incorrect view” class for the $v_i$-th viewpoint, indicates how likely it is that the estimated viewpoint variable $v_i$ is incorrect.

Based on the above discussion, we substantiate (1) as follows. Letting $P_i = \{p_{j,k}^{(i)}\} \in \mathbb{R}_+^{M \times (N+1)}$ denote a matrix composed of $P(\hat{y}_i | x_i, v_j)$ for all the $M$ viewpoints and $N + 1$ classes, the target value of $P_i$ in the case that $v_i$ is correctly estimated is defined as follows:

$$p_{j,k}^{(i)} = \begin{cases} 1 & (j = v_i \text{ and } k = y) \text{ or } (j \neq v_i \text{ and } k = N + 1) \\ 0 & \text{(otherwise)} \end{cases}$$

In this way, (1) can be rewritten as the following cross-entropy optimization problem:

$$\max_{R, \{v_i\}_{i=1}^M} \sum_{i=1}^M \left( \log p_{v_i,y}^{(i)} + \sum_{j \neq v_i} \log p_{j,N+1}^{(i)} \right).$$
If we fix $\{v_i\}_{i=1}^M$ here, the above can be written as a sub-
problem of optimizing $R$ as follows:

$$
\max_R \sum_{i=1}^M \left( \log p_{v_i,y}^{(i)} + \sum_{j \neq v_i} \log p_{v_j,N+1}^{(i)} \right),
$$

(4)

where the parameters of $R$ can be iteratively updated via
standard back-propagation of $M$ softmax losses. Since $\{v_i\}_{i=1}^M$ are not constant but latent variables that need to
be optimized during the training of $R$, we employ alternat-
ning optimization of $R$ and $\{v_i\}_{i=1}^M$. More specifically, in
every iteration, our method determines $\{v_i\}_{i=1}^M$ according
to $P_t$ obtained via forwarding of (fixed) $R$, and then update
$R$ according to the estimated $\{v_i\}_{i=1}^M$ by fixing them.

The latent viewpoint variables $\{v_i\}_{i=1}^M$ are determined by solving the following problem:

$$
\max_{\{v_i\}_{i=1}^M} \sum_{i=1}^M \left( \log p_{v_i,y}^{(i)} + \sum_{j \neq v_i} \log p_{v_j,N+1}^{(i)} \right)
= \max_{\{v_i\}_{i=1}^M} \sum_{i=1}^M \left( \log p_{v_i,y}^{(i)} + \sum_{j=1}^M \log p_{v_j,N+1}^{(i)} - \log p_{v_i,N+1}^{(i)} \right)
= \max_{\{v_i\}_{i=1}^M} \prod_{i=1}^M \frac{p_{v_i,y}^{(i)}}{p_{v_i,N+1}^{(i)}},
$$

(5)

in which the conversion used the fact that $\sum_{j=1}^M \log p_{v_j,N+1}^{(i)}$ is constant w.r.t. $\{v_i\}_{i=1}^M$. Because the number of can-
didates for $\{v_i\}_{i=1}^M$ is limited, we calculate the evalua-
tion value of $\{v_i\}_{i=1}^M$ for all the candidates and take the best
choice. The decision of $\{v_i\}_{i=1}^M$ in this way emphasizes
view-specific features for object categorization, which con-
tributes to the self-configuration of objects in the dataset.

In the inference phase, RotationNet takes as input $M'$
$\{v_i\}_{i=1}^M$ images of a test object instance, either si-
multaneously or sequentially, and outputs $M'$ probabilities.
Finally, it integrates the $M'$ outputs to estimate the cate-
ogy of the object and the viewpoint variables as follows:

$$
\{\hat{y}, \hat{v}_i\}_{i=1}^{M'} = \arg \max_{y, \{v_i\}_{i=1}^{M'}} \prod_{i=1}^{M'} \frac{p_{v_i,y}^{(i)}}{p_{v_i,N+1}^{(i)}}.
$$

(6)

Similarly to the training phase, we decide $\{\hat{v}_i\}_{i=1}^{M'}$ accord-
ing to the outputs $\{P_t\}_{i=1}^{M'}$. Thus RotationNet is able to
estimate the pose of the object as well as its category label.

**Viewpoint setups for training** While choices of the
viewpoint variables $\{v_i\}_{i=1}^{M'}$ can be arbitrary, we consider
two setups in this paper, with and without an upright ori-
entation assumption, similarly to MVCNN [34]. The for-
mer case is often useful with images of real objects captured
with one-dimensional turning tables, whereas the latter case
is rather suitable for unaligned 3D models. We also consider
the third case that is also based on the upright orientation
assumption (as the first case) but with multiple elevation
levels. We illustrate the three viewpoint setups in Fig. 3.

**Case (i): with upright orientation** In the case where we
assume upright orientation, we fix a specific axis as the ro-
tation axis (e.g., the $z$-axis), which defines the upright ori-
entation, and then place viewpoints at intervals of the angle
$\theta$ around the axis, elevated by $\phi$ (set to $30^\circ$ in this paper)
from the ground plane. We set $\theta = 30^\circ$ in default, which
yields 12 views for an object ($M = 12$). We define that
"view $m+1$" is obtained by rotating the view position "view $m"
by the angle $\theta$ about the $z$-axis. Note that the view ob-
tained by rotating "view $m'$" by the angle $\theta$ about the $z$-axis
corresponds to "view 1." We assume the sequence of in-
put images is consistent with respect to a certain direction
of rotation in the training phase. For instance, if $v_i$ is $m$
($m < M$), then $v_{i+1}$ is $m + 1$. Thus the number of can-
didates for all the viewpoint variables $\{v_i\}_{i=1}^M$ is $M$.

**Case (ii): w/o upright orientation** In the case where we
do not assume upright orientation, we place virtual cameras
on the $M = 20$ vertices of a dodecahedron encompassing
the object. This is because a dodecahedron has the largest
number of vertices among regular polyhedra, where view-
points can be completely equally distributed in 3D space.
Unlike case (i), where there is a unique rotation direction,
there are three different patterns of rotation from a certain
view, because three edges are connected to each vertex of a
dodecahedron. Therefore, the number of candidates for all
the viewpoint variables $\{v_i\}_{i=1}^M$ is $60 (= 3M^2)$.

**Case (iii): with upright orientation and multiple eleva-
tion levels** This case is an extension of case (i). Un-
like case (i) where the elevation angle is fixed, we place
virtual cameras at intervals of $\phi$ in $[90^\circ, 90^\circ]$. There are
$M = M_a \times M_e$ viewpoints, where $M_a = \frac{360^\circ}{\phi}$ and
$M_e = \frac{180^\circ}{\phi} + 1$. As with the case (i), the number of can-
didates for all the viewpoint variables $\{v_i\}_{i=1}^M$ is $M_a$ due to
the upright orientation assumption.

---

$^1$A dodecahedron has 60 orientation-preserving symmetries.
4. Experiments

In this section, we show the results of the experiments with 3D model benchmark datasets (Sec. 4.1), a real image benchmark dataset captured with a one-dimensional turning table (Sec. 4.2), and our new dataset consisting of multi-view real images of objects viewed with two rotational degrees of freedom (Sec. 4.3). The baseline architecture of our CNN is based on AlexNet [16], which is smaller than the VGG-M network architecture that MVCNN [34] used. To train RotationNet, we fine-tune the weights pre-trained using the ILSVRC 2012 dataset [27]. We used classical momentum SGD with a learning rate of 0.0005 and a momentum of 0.9 for optimization.

As a baseline method, we also fine-tuned the pre-trained weights of a standard AlexNet CNN that only predicts object categories. To aggregate the predictions of multi-view images, we summed up all the scores obtained through the CNN. This method can be recognized as a modified version of MVCNN [34], where the view-pooling layer is placed after the final softmax layer. We chose average pooling for the view-pooling layer in this setting of the baseline, because we observed that the performance was better than that with max pooling. We also implemented MVCNN [34] based on the AlexNet architecture with the original view-pooling layer for a fair comparison.

4.1. Experiment on 3D model datasets

We first describe the experimental results on two 3D model benchmark datasets, ModelNet10 and ModelNet40 [39]. ModelNet10 consists of 4,899 object instances in 10 categories, whereas ModelNet40 consists of 12,311 object instances in 40 categories. First, we show the change of object classification accuracy versus the number of views used for prediction in cases (i) and (ii) with ModelNet40 and ModelNet10, respectively, in Figs. 4(a)-(b) and Fig. 4(d)-(e). For fair comparison, we used the same training and test split of ModelNet40 as in [39] and [54]. We prepared multi-view images (i) with the upright orientation assumption and (ii) without the upright orientation assumption using the rendering software published in [34]. Here, we show the average scores of 120 trials with randomly selected multi-view sets. In Figs. 4(a) and 4(d), which show the results with ModelNet40, we also draw the scores with the original MVCNN using Support Vector Machine (SVM) reported in [34]. Interestingly, as we focus on the object classification task whereas Su et al. [34] focused more on object retrieval task, we found that the baseline method with late view-pooling is slightly better in this case than the original MVCNN with the view-pooling layer in the middle. The baseline method does especially well with ModelNet10 in case (i) (Fig. 4(b)), where it achieves the best performance among the methods. With ModelNet40 in case (i) (Fig. 4(a)), RotationNet achieved a comparable result with MVCNN when we used all the 12 views as input. In case (ii) (Figs. 4(d) and (e)), where we consider full 3D rotation, RotationNet demonstrated superior performance to other methods. Only with three views, it showed comparable performance to that of MVCNN with a full set (80 views) of multi-view images.

Next, we investigate the performance of RotationNet with three different architectures: AlexNet [16], VGG-M [6], and ResNet-50 [12]. Table 1 shows the classification accuracy on ModelNet40 and ModelNet10. Because we deal with discrete viewpoints, we altered 11 different camera system orientations (similarly to [8]) and calculated the mean and maximum accuracy of those trials. Surprisingly, the performance difference among different architectures is marginal compared to the difference caused by different camera system orientations. It indicates that the placement of viewpoints is the most important factor in multi-view-based 3D object classification.

Finally, we summarize the comparison of classification accuracy on ModelNet40 and ModelNet10 to existing 3D object classification methods in Table 5. RotationNet (with VGG-M architecture) significantly outperformed existing methods with both the ModelNet40 and ModelNet10 datasets. We reported the maximum accuracy among the aforementioned 11 rotation trials. Note that the average accuracy of those trials on ModelNet40 was 94.68%, which is still superior to the current state-of-the-art score 93.8% reported by Wang et al. [57]. Besides, Wang et al. [57] used additional feature modalities: surface normals and normalized depth values to improve the performance by > 1%.

4.2. Experiment on a real image benchmark dataset

Next, we describe the experimental results on a benchmark RGBD dataset published in [18], which consists of real images of objects on a one-dimensional rotation table. This dataset contains 300 object instances in 51 categories. Although it contains depth images and 3D point clouds, we used only RGB images in our experiment. We applied the upright orientation assumption (case (i)) in this

<table>
<thead>
<tr>
<th>Archit.</th>
<th>ModelNet40 Mean</th>
<th>Max</th>
<th>ModelNet10 Mean</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>93.70 ± 1.07</td>
<td>96.39</td>
<td>94.52 ± 1.01</td>
<td>97.58</td>
</tr>
<tr>
<td>VGG-M</td>
<td>94.68 ± 1.16</td>
<td>97.37</td>
<td>94.82 ± 1.17</td>
<td>98.46</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>94.77 ± 1.10</td>
<td>96.92</td>
<td>94.80 ± 0.96</td>
<td>97.80</td>
</tr>
</tbody>
</table>

Table 1. Comparison of classification accuracy (%) with RotationNet based on different architectures.

---

2We do not include the scores of “VRN Ensemble” [5] using ensembling technique because is written in [5] “we suspect that this result is not general, and do not claim it with our main results.” The reported scores are 95.54% with ModelNet40 and 97.14% with ModelNet10, which are both outperformed by RotationNet with any architecture (see Table 1).
Figure 4. Classification accuracy vs. number of views used for prediction. From left to right are shown the results on ModelNet40, ModelNet10, and our new dataset MIRO. The results in case (i) are shown in top and those in case (ii) are shown in bottom. See Table 5 for an overall performance comparison to existing methods on ModelNet40 and ModelNet10.

Table 2. Accuracy of classification and viewpoint estimation (%) in case (i) with RGBD.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>class</th>
<th>view</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVCNN (softmax)</td>
<td>86.08</td>
<td>-</td>
</tr>
<tr>
<td>Baseline</td>
<td>88.73</td>
<td>-</td>
</tr>
<tr>
<td>Fine-grained, $T=300$</td>
<td>81.23</td>
<td>26.94</td>
</tr>
<tr>
<td>Fine-grained, $T=4K$</td>
<td>76.95</td>
<td>31.96</td>
</tr>
<tr>
<td>RotationNet</td>
<td>89.31</td>
<td>33.59</td>
</tr>
</tbody>
</table>

Table 3. Accuracy of classification and viewpoint estimation (%) in case (i) with MIRO.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>class</th>
<th>view</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVCNN (softmax)</td>
<td>94.17</td>
<td>-</td>
</tr>
<tr>
<td>Baseline</td>
<td>95</td>
<td>-</td>
</tr>
<tr>
<td>Fine-grained, $T=800$</td>
<td>92.76</td>
<td>56.72</td>
</tr>
<tr>
<td>Fine-grained, $T=4K$</td>
<td>91.35</td>
<td>58.33</td>
</tr>
<tr>
<td>RotationNet</td>
<td>98.33</td>
<td>85.83</td>
</tr>
</tbody>
</table>

Table 4. Accuracy of classification and viewpoint estimation (%) in case (ii) with MIRO.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>class</th>
<th>view</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVCNN (softmax)</td>
<td>95</td>
<td>-</td>
</tr>
<tr>
<td>Baseline</td>
<td>95.83</td>
<td>-</td>
</tr>
<tr>
<td>Fine-grained, $T=1.1K$</td>
<td>94.21</td>
<td>70.63</td>
</tr>
<tr>
<td>Fine-grained, $T=2.6K$</td>
<td>93.54</td>
<td>72.38</td>
</tr>
<tr>
<td>RotationNet</td>
<td>99.17</td>
<td>75.67</td>
</tr>
</tbody>
</table>

Table 2 summarizes the classification and viewpoint estimation accuracies. The baseline method and MVCNN are not able to estimate viewpoints because they are essentially viewpoint invariant. As another baseline approach to compare, we learned a CNN with AlexNet architecture that outputs $612 (= 51 \times 12)$ scores to distinguish both viewpoints and categories, which we call “Fine-grained.” Here, $T$ denotes the number of iterations that the CNN parameters are updated in the training phase. As shown in Table 2, the classification accuracy with “Fine-grained” decreases while its viewpoint estimation accuracy improves as the iteration grows. We consider this is because the “Fine-grained” classifiers become more and more sensitive to intra-class appearance variation through training, which affects the categorization accuracy. In contrast, RotationNet demonstrated the best performance in both object classification and viewpoint estimation, although the ground-truth poses are not given to RotationNet during the training.
We describe the experimental results on our new dataset “Multi-view Images of Rotated Objects (MIRO)” in this section. We used Ortery’s 3D MFP studio3 to capture multi-view images of objects with 3D rotations. The RGBD benchmark dataset [18] has two issues for training multiview based CNNs: insufficient number of object instances per category (which is a minimum of two for training) and inconsistent cases to the upright orientation assumption. There are several cases where the upright orientation assumption is actually invalid; the attitudes of object instances against the rotation axis are inconsistent in some object categories. Also, this dataset does not include the bottom faces of objects on the turning table. Our MIRO dataset includes 10 object instances per object category. It consists of 120 object instances in 12 categories in total. We captured each object instance with $M_e = 10$ levels of elevation angles and 16 levels of azimuth angles to obtain 160 images. For our experiments, we used 16 images ($\theta = 22.5^\circ$) with 0$^\circ$ elevation of an object instance in case (i). We carefully captured all the object instances in each category to have the same upright direction in order to evaluate performance in the case (i). For case (ii), we used 20 images observed from the 20 vertices of a dodecahedron encompassing an object.

Finally, we demonstrate the performance of RotationNet for real-world applications. For training, we used our MIRO dataset with the viewpoint setup case (iii), where all the outputs for images with 10 levels of elevation angles are concatenated, which enables RotationNet to distinguish both viewpoints and categories, and the overall results are summarized in Tables 3 and 4. Similar to the results with an RGBD dataset described above, there is a trade-off between object classification and viewpoint estimation accuracies in the “Fine-grained” approach. RotationNet achieved the best performance in both object classification and viewpoint estimation, which demonstrates the strength of the proposed approach.

Table 6. Comparison on object instance/category recognition and pose estimation on RGBD dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ModelNet40</th>
<th>ModelNet10</th>
</tr>
</thead>
<tbody>
<tr>
<td>RotationNet</td>
<td>97.37</td>
<td>98.46</td>
</tr>
<tr>
<td>Dominant Set Clustering</td>
<td>93.8</td>
<td>-</td>
</tr>
<tr>
<td>KD-Networks</td>
<td>91.8</td>
<td>94.0</td>
</tr>
<tr>
<td>MVCNN-MultiRes</td>
<td>91.4</td>
<td>-</td>
</tr>
<tr>
<td>ORION</td>
<td>91.33</td>
<td>93.61</td>
</tr>
<tr>
<td>VRN</td>
<td>90.80</td>
<td>93.11</td>
</tr>
<tr>
<td>FusionNet</td>
<td>90.70</td>
<td>92.80</td>
</tr>
<tr>
<td>Pairwise</td>
<td>90.7</td>
<td>91.1</td>
</tr>
<tr>
<td>PANORAMA-NN</td>
<td>90.10</td>
<td>-</td>
</tr>
<tr>
<td>MVCNN</td>
<td>88.4</td>
<td>-</td>
</tr>
<tr>
<td>Set-convolution</td>
<td>87.8</td>
<td>91.5</td>
</tr>
<tr>
<td>DeepPano</td>
<td>86.90</td>
<td>93.39</td>
</tr>
<tr>
<td>PointNet</td>
<td>86.2</td>
<td>-</td>
</tr>
<tr>
<td>Geometry Image</td>
<td>83.9</td>
<td>88.4</td>
</tr>
<tr>
<td>3D-GAN</td>
<td>83.30</td>
<td>91.00</td>
</tr>
<tr>
<td>ECC</td>
<td>83.2</td>
<td>-</td>
</tr>
<tr>
<td>GIFT</td>
<td>83.10</td>
<td>92.35</td>
</tr>
<tr>
<td>VoxNet</td>
<td>83</td>
<td>92</td>
</tr>
<tr>
<td>Beam Search</td>
<td>81.26</td>
<td>88</td>
</tr>
<tr>
<td>DeepPano</td>
<td>77.63</td>
<td>85.45</td>
</tr>
<tr>
<td>3DShapeNets</td>
<td>77</td>
<td>83.50</td>
</tr>
<tr>
<td>PointNet</td>
<td>77.6</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5. Comparison of classification accuracy (%). RotationNet achieved the state-of-the-art performance both with ModelNet40 and ModelNet10.

Table 6. Comparison on object instance/category recognition and pose estimation (rad) on 12 objects with 12 views.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Instance (%)</th>
<th>Category (%)</th>
<th>Avg. Pose (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lai et al. [19]</td>
<td>78.40</td>
<td>94.30</td>
<td>53.50</td>
</tr>
<tr>
<td>Zhang et al. [42]</td>
<td>74.79</td>
<td>93.10</td>
<td>61.57</td>
</tr>
<tr>
<td>Bakry et al. [2]</td>
<td>80.10</td>
<td>94.84</td>
<td>76.63</td>
</tr>
<tr>
<td>Elhoseiny et al. [9]</td>
<td>-</td>
<td>97.14</td>
<td>79.30</td>
</tr>
<tr>
<td>Ours - single view</td>
<td>90.44</td>
<td>96.55</td>
<td>78.67</td>
</tr>
<tr>
<td>Ours - 12 views</td>
<td>97.45</td>
<td>99.51</td>
<td>81.17</td>
</tr>
</tbody>
</table>

4.3. Experiment on a 3D rotated real image dataset

We describe the experimental results on our new dataset “Multi-view Images of Rotated Objects (MIRO)” in this section. We used Ortery’s 3D MFP studio3 to capture multi-

3https://www.ortery.com/photography-equipment/3d-modeling/
improved by using multiple views.

5. Discussion

We proposed RotationNet, which jointly estimates object category and viewpoint from each single-view image and aggregates the object class predictions obtained from a partial set of multi-view images. In our method, object instances are automatically aligned in an unsupervised manner with both inter-class and intra-class structures based on their appearance during the training. In the experiment using 3D object benchmark datasets ModelNet40 and ModelNet10, RotationNet significantly outperformed the state-of-the-art methods based on voxels, point clouds, and multi-view images. RotationNet is also able to achieve comparable performance to MVCNN [34] with 80 different multi-view images using only a couple of view images, which is important for real-world applications. Another contribution is that we developed a publicly available new dataset named MIRO. Using this dataset and RGBD object benchmark dataset [18], we showed that RotationNet even outperformed supervised learning based approaches in a pose estimation task. We consider that our pose estimation performance benefits from view-specific appearance information shared across classes due to the inter-class self-alignment.

Similar to MVCNN [34] and any other 3D object classification method that considers discrete variance of rotation, RotationNet has the limitation that each image should be observed from one of the pre-defined viewpoints. The discrete pose estimation by RotationNet, however, demonstrated superior performance to existing methods on the RGBD object benchmark dataset. It can be further improved by introducing a fine pose alignment post-process using e.g., iterative closest point (ICP) algorithm. Another potential avenue to look into is the automatic selection of the best camera system orientations, since it has an effect on object classification accuracy.

Acknowledgment

This project is supported by the New Energy and Industrial Technology Development Organization (NEDO). The authors would like to thank Hiroki Matsuno for his support in developing the HoloLens application.
References


