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

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.

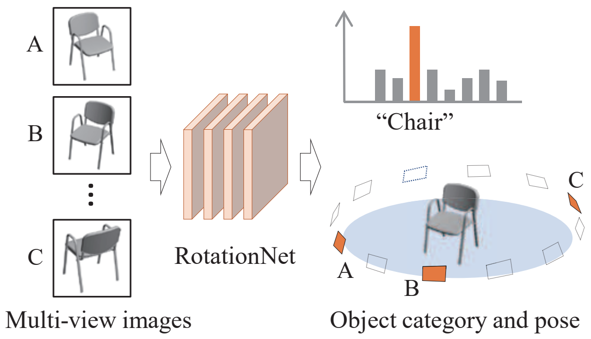

Figure 1. Illustration of the proposed method *RotationNet*. RotationNet takes a partial set ($\geq$ 1 images) of the full multi-view images of an object as input and predicts its object category by rotation, where the best pose is selected to maximize the object category likelihood. Here, viewpoints from which the images are observed are jointly estimated to predict the pose of the object.

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 innately 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 view 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.

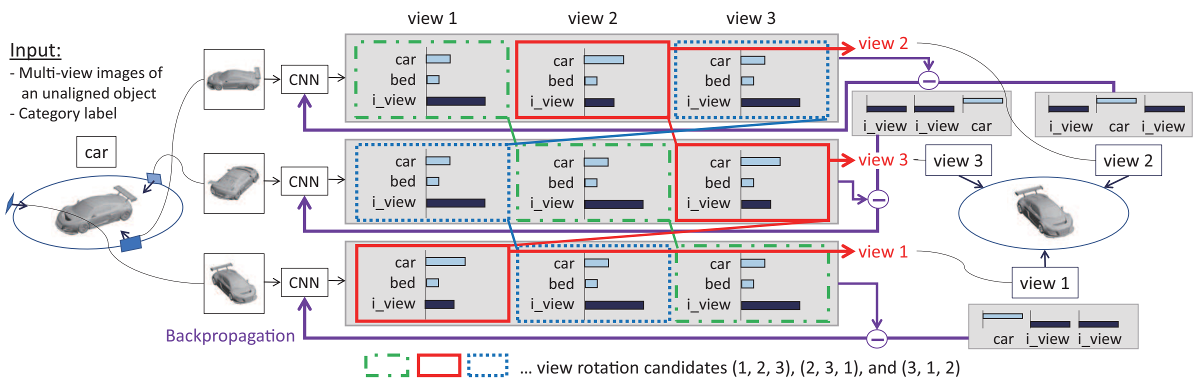

Figure 2. Illustration of the training process of RotationNet, where the number of views M is 3 and the number of categories N is 2. A training sample consists of M images of an unaligned object and its category label y. For each input image, our CNN (RotationNet) outputs M histograms with $N+1$ bins 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 image corresponds to views 1, 2, and 3. There are three candidates for view rotation: (1, 2, 3), (2, 3, 1), and (3, 1, 2). For each candidate, we calculate the score for the ground-truth category ("car" in this case) by multiplying the histograms and selecting the best choice: (2, 3, 1) in this case. Finally, we update the CNN parameters in a standard back-propagation manner with the estimated viewpoint variables. Note that it is the same CNN that is being used.

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 $\{\boldsymbol{x}_i\}_{i=1}^M$ and its category label $y \in \{1, \ldots, N\}$. We attach a viewpoint variable $v_i \in \{1, \ldots, M\}$ to each image $\boldsymbol{x}_i$ 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 $P(\hat{y}_i \mid \boldsymbol{x}_i, v_i = j)$ where $j \in \{1, \ldots, M\}$ for each image $\boldsymbol{x}_i$. Here, $\hat{y}_i$ denotes an estimate of the object category label for $\boldsymbol{x}_i$. For the training of RotationNet, we input the set of images $\{\boldsymbol{x}_i\}_{i=1}^M$ simultaneously and solve the following optimization problem:

$$\max_{R, \{v_i\}_{i=1}^M} \prod_{i=1}^{M} P(\hat{y}_i = y \mid \boldsymbol{x}_i, v_i). \tag{1}$$

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 $\{\boldsymbol{x}_i\}_{i=1}^M$.

Now, we describe how we design $P(\hat{y}_i \mid \boldsymbol{x}_i, v_i)$ outputs. First of all, the category likelihood $P(\hat{y}_i = y \mid \boldsymbol{x}_i, v_i)$ should become close to one when the estimated v_i is correct; in other words, the image $\boldsymbol{x}_i$ is truly captured from the v_i-th viewpoint. Otherwise, in the case that the estimated v_i is incorrect, $P(\hat{y}_i = y \mid \boldsymbol{x}_i, v_i)$ may not necessarily be high because the image $\boldsymbol{x}_i$ is captured from a different viewpoint. As described above, we decide the viewpoint variables $\{v_i\}_{i=1}^M$ according to the $P(\hat{y}_i = y \mid \boldsymbol{x}_i, v_i)$ 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 $P(\hat{y}_i \mid \boldsymbol{x}_i, v_i)$ by applying softmax functions to the $(N+1)$-dimensional outputs, where $\sum_{\hat{y}_i=1}^{N+1} P(\hat{y}_i \mid \boldsymbol{x}_i, v_i) = 1$. Note that $P(\hat{y}_i = N+1 \mid \boldsymbol{x}_i, v_i)$, which corresponds to the probability that the image $\boldsymbol{x}_i$ 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 $P_i = \left[p_{j,k}^{(i)}\right] \in \mathbb{R}_+^{M \times (N+1)}$ denote a matrix composed of $P(\hat{y}_i \mid \boldsymbol{x}_i, v_i)$ 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:

$$p_{j,k}^{(i)} = \begin{cases} 1 & (j = v_i \text{ and } k = y) \text{ or } (j \neq v_i \text{ and } k = N+1) \\ 0 & (\text{otherwise}). \end{cases} \tag{2}$$

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

$$\max_{R, \{v_i\}_{i=1}^M} \sum_{i=1}^{M} \left(\log p_{v_i,y}^{(i)} + \sum_{j \neq v_i} \log p_{j,N+1}^{(i)} \right). \tag{3}$$

If we fix $\{v_i\}_{i=1}^M$ here, the above can be written as a sub-problem of optimizing R as follows:

$$\max_R \sum_{i=1}^M \left(\log p_{v_i,y}^{(i)} + \sum_{j \neq v_i} \log p_{j,N+1}^{(i)} \right), \quad (4)$$

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 alternating 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_i 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:

$$\begin{aligned}
&\max_{\{v_i\}_{i=1}^M} \sum_{i=1}^M \left(\log p_{v_i,y}^{(i)} + \sum_{j \neq v_i} \log p_{j,N+1}^{(i)} \right) \\
&= \max_{\{v_i\}_{i=1}^M} \sum_{i=1}^M \left(\log p_{v_i,y}^{(i)} + \sum_{j=1}^M \log p_{j,N+1}^{(i)} - \log p_{v_i,N+1}^{(i)} \right) \\
&= \max_{\{v_i\}_{i=1}^M} \prod_{i=1}^M \frac{p_{v_i,y}^{(i)}}{p_{v_i,N+1}^{(i)}}, \quad (5)
\end{aligned}$$

in which the conversion used the fact that $\sum_{j=1}^M \log p_{j,N+1}^{(i)}$ is constant w.r.t. $\{v_i\}_{i=1}^M$. Because the number of candidates for $\{v_i\}_{i=1}^M$ is limited, we calculate the evaluation value of (5) 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 contributes to the self-alignment of objects in the dataset.

In the inference phase, RotationNet takes as input M' ($1 \leq M' \leq M$) images of a test object instance, either simultaneously or sequentially, and outputs M' probabilities. Finally, it integrates the M' outputs to estimate the category of the object and the viewpoint variables as follows:

$$\left\{ \hat{y}, \{\hat{v}_i\}_{i=1}^{M'} \right\} = \arg\max_{y,\{v_i\}_{i=1}^{M'}} \prod_{i=1}^{M'} \frac{p_{v_i,y}^{(i)}}{p_{v_i,N+1}^{(i)}}. \quad (6)$$

Similarly to the training phase, we decide $\{\hat{v}_i\}_{i=1}^{M'}$ according to the outputs $\{P_i\}_{i=1}^{M'}$. 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 orientation assumption, similarly to MVCNN [34]. The former 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.

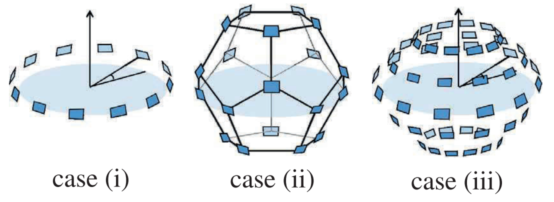

Figure 3. Illustration of three viewpoint setups considered in this work. A target object is placed on the center of each circle.

Case (i): with upright orientation In the case where we assume upright orientation, we fix a specific axis as the rotation axis (*e.g.*, the z-axis), which defines the upright orientation, and then place viewpoints at intervals of the angle θ around the axis, elevated by ϕ (set to 30° 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 θ about the z-axis. Note that the view obtained by rotating "view M" by the angle θ about the z-axis corresponds to "view 1." We assume the sequence of input 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 candidates 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 viewpoints 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 ($= 3M$)[1].

Case (iii): with upright orientation and multiple elevation levels This case is an extension of case (i). Unlike case (i) where the elevation angle is fixed, we place virtual cameras at intervals of ϕ in $[-90^\circ, 90^\circ]$. There are $M = M_a \times M_e$ viewpoints, where $M_a = \frac{360^\circ}{\theta}$ and $M_e = \frac{180^\circ}{\phi} + 1$. As with the case (i), the number of candidates 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.