Joint Object Category and 3D Pose Estimation from 2D Images

Siddharth Mahendran siddharthm@jhu.edu
Haider Ali hali@jhu.edu
René Vidal rvidal@cis.jhu.edu
Center for Imaging Science, Johns Hopkins University

Abstract

2D object detection is the task of finding (i) what objects are present in an image and (ii) where they are located, while 3D pose estimation is the task of finding the pose of these objects in 3D space. State-of-the-art methods for solving these tasks follow a two-stage approach where a 3D pose estimation system is applied to bounding boxes (with associated category labels) returned by a 2D detection method. This paper addresses the task of joint object category and 3D pose estimation given a 2D bounding box. We design a residual network based architecture to solve these two seemingly orthogonal tasks with new category-dependent pose outputs and loss functions, and show state-of-the-art performance on the challenging Pascal3D+ dataset.

1. Introduction

Scene understanding is a core problem in computer vision and an important part of many modern vision challenges. One way to understand the image of a scene is to describe it in terms of the objects present in it. This involves answering two key questions:

Q1. What objects are present in the image?

Q2. Where are these objects located in the image?

Modern vision systems usually answer these questions jointly in the task of object recognition. They return 2D bounding boxes around the objects in an image, also known as object detection (Q2), and associate each bounding box with an object category label, also known as object category estimation (Q1). However, this 2D description of the world is very limited, especially for tasks like autonomous driving and robot manipulation which require a 3D understanding of the scene. This requires us to also answer the question:

Q3. What are the 3D poses of the objects present in the image?

otherwise known as the task of 3D pose estimation.

3D pose estimation is a very old and fundamental problem in the computer vision and robotics community. With the recent successes of deep learning for the tasks of image classification and 2D object detection, many recent works like [35, 33, 37, 25] have used convolutional neural networks for pose estimation. Howerver, all these works use the output of 2D object detection systems like [11] and [31] as input to their 3D pose estimation system. Effectively, they are answering Q1 and Q2 first, and then Q3 in a pipeline approach. Ideally, we would like to have a system that answers all three questions simultaneously i.e a system that does 3D pose estimation for every object it detects, analogous to Mask R-CNN [13] which combines object detection with instance segmentation.

Paper contributions: In this paper, we take a step in that direction by combining the tasks of object category estimation (Q1) and 3D pose estimation (Q3), assuming we have bounding boxes around objects in the image (Q2). This is similar to the original R-CNN work [11] where a proposal algorithm would return candidate bounding boxes which would then be refined and classified. We assume the bounding box is provided by an oracle and we predict the object category label and 3D pose, see Fig. 1.

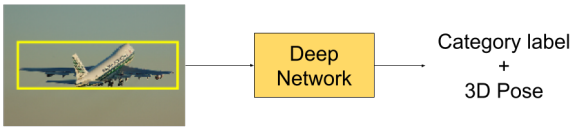

Figure 1. Overview of our problem formulation: Given a 2D image and a bounding box around an object in the image, return the object's 3D pose and category label

Our key contributions are: (1) a residual network based architecture to solve the tasks of object category estimation and 3D pose estimation jointly, (2) new formulations and loss functions for 3D pose when the object category label is not known and (3) state-of-the-art results on the challenging Pascal3D+ [38] dataset. We also present an analysis of various decision choices made in our proposed approach. To the best of our knowledge, we are the first to use residual networks [14], that have worked very well for the task of image classification, for 3D pose estimation. We do note

that [12] also use residual networks but for azimuth / orientation estimation.

2. Related Work

There are two main lines of research that are relevant to our work: (i) 3D Pose estimation given object category label and (ii) Joint object category and pose estimation. There are many non-deep learning based approaches like [23, 16, 4, 30, 32, 5, 1, 17], and many others that have not been mentioned here, that have designed systems to solve these two tasks. However, due to space constraint we restrict our discussion to deep learning based approaches.

3D Pose estimation given object category label: Current literature for this task can be grouped into two categories: (1) predict 2D keypoints from images and then align 3D models with these keypoints to recover 3D pose and (2) predict 3D pose directly from 2D images. The first category includes methods like Pavlakos *et al*. [29] and Wu *et al*. [37] which recover a full 6-dimensional pose (azimuth, elevation, camera-tilt, depth, image-translation). Both methods train on images that have been annotated with 2D keypoints that correspond to semantic keypoints on 3D object models. Given a 2D image, they first generate a probabilistic heatmap of 2D keypoints and then recover 3D pose by aligning these 2D keypoints with the 3D keypoints. The second category includes methods like Tulsiani and Malik [35], Su *et al*. [33], Mahendran *et al*. [25] and Wang *et al*. [36], where they recover the 3D rotation between object and camera which corresponds to a 3-dimensional pose (azimuth, elevation, camera-tilt). [35] and [33] setup a pose-classification problem by discretizing the three angles into pose-labels and minimize the cross-entropy loss during training. [25] and [36] on the other hand, solve a pose regression problem. While [36] directly regresses the three angles with mean squared loss, [25] uses axis-angle or quaternion representations of 3D rotations and minimizes a geodesic loss during training. Our work is closest to [25] in that we also use an axis-angle representation and geodesic loss function while solving a 3D pose regression problem. However, a key difference between our work and [25] is that our network architecture is based on residual networks [14, 15] whereas [25] use VGG-M [7] as their base network. The other big difference is of course that we are trying to solve a much more challenging task of joint category and pose estimation. [22] does a good job of reviewing current literature on 3D pose estimation using deep networks.

Joint object category and pose estimation: Braun *et al*. [6] and Massa *et al*. [26, 27] work on the problem of joint object detection and pose estimation. Elhoseiny *et al*. [10] explicitly work on the problem of joint object category and pose estimation. However in all these works, the pose is restricted to just the azimuth or yaw angle. We, on the other hand, recover the full 3D rotation matrix which is a much harder problem. Also, these works all consider pose to be independent of category *i.e*. they set up a multi-task network that has separate branches for pose and category label, which are then treated independently. We design our pose network to be category-dependent *i.e*. there is a pose output associated with each object category, and our category network output determines final pose output.

Paper Outline: We first describe our model for joint object category and 3D pose estimation in §3 specifically our network architecture §3.1 and the new category-dependent loss functions we use for the joint task §3.2. We then describe our experimental evaluation and analysis in §4.

3. Joint object category and pose estimation

3.1. Network architecture

At a high level, our network architecture is an extension of the architecture proposed in [25] (shown in Fig. 2) which consists of two parts: a feature network shared between all object categories and a pose network for each object category. Their overall network takes as input both an image and the ground-truth object category label, and selects the output of the pose network corresponding to the correct category as the final pose.

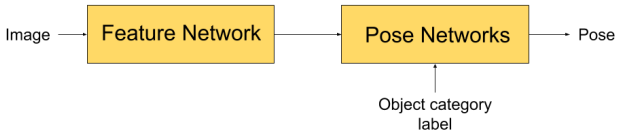

Figure 2. Network architecture for 3D pose estimation with known object category label

In this paper, we consider the more challenging situation in which the object category is unknown. Therefore, our network architecture also includes a category network that takes as input the features returned by the feature network to estimate the object category label (shown in Fig. 3). The output of the category network is then used as an input to the pose network to detemine the final pose output. Another key difference is that we use a residual network ResNet-50 [14] as our feature network. To be more precise, in our experiments, we use the ResNet-50 upto stage-4 as our feature network, ResNet-50 stage-5 as our category network and the 3-layer pose networks in [25] as our pose networks.

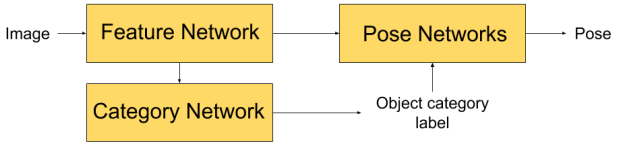

Figure 3. Network architecture for joint object category and 3D pose estimation

3.2. Loss functions and Representations

Our network returns the 3D pose, a rotation matrix R, and object category label c. In general, a loss function between ground-truth pose R^*, ground-truth category label c^* and our network output (R, c), can be expressed as $\mathcal{L}(R, c, R^*, c^*)$ and one simple choice of the overall loss is to define it as a sum of the pose loss and category loss *i.e.* $\mathcal{L}(R, c, R^*, c^*) = \mathcal{L}_{pose}(R(c), R^*) + \lambda \mathcal{L}_{category}(c, c^*)$. We use the standard categorical cross-entropy loss for our category loss. Note that the way we have depicted our pose loss $\mathcal{L}_{pose}(R(c), R^*)$, it explicitly encodes the fact that our pose output depends on the estimated category.

The pose loss depends on how we represent a rotation matrix and we use the axis-angle representation, $R = \text{expm}(\theta[v]_\times)$, where v corresponds to the axis of rotation and θ is the angle of rotation. $[v]_\times$ is the skew-symmetric matrix generated from vector $v = [v_1, v_2, v_3]^T$, *i.e.* $[v]_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}$. By restricting $\theta \in [0, \pi)$, we create a one-to-one correspondence between rotation R and axis-angle vector $y = \theta v$. This allows us to use a geodesic loss function [24] on the space of rotation matrices $\mathcal{L}(R_1, R_2)$ as our pose loss $\mathcal{L}_{pose}$.

$$\mathcal{L}(R_1, R_2) = \frac{\| \log(R_1 R_2^T) \|_F}{\sqrt{2}} \tag{1}$$

The pose loss between two axis-angle vectors y_1 and y_2 is now defined as $\mathcal{L}_p(y_1, y_2) \equiv \mathcal{L}(R_1, R_2)$ where R_1 and R_2 are their respective rotation matrices.

Let y_i be the output of i-th pose network. When the object category is known, we can choose the pose output corresponding to the correct category label *i.e.* with ground-truth object category label c^*, the pose output is y_{c^*} and then compare it with the true pose y^*, *i.e.*

$$\mathcal{L}_p(y^*, \{y_i\}|c^*) = \mathcal{L}_p(y^*, y_{c^*}) \tag{2}$$

When we do not have the ground-truth object category label, we can estimate the pose output in two different ways which leads to the following two loss functions.

Weighted loss function: Assuming the output of the category network is a probability vector whose i-th entry $p(c = i)$ is the probability that the image corresponds to category i, the estimated pose is given by $y_{wgt}(c) = \sum_i y_i p(c = i)$ leading to the pose loss

$$\mathcal{L}_p(y^*, y(c)) = \mathcal{L}_p(y^*, \sum_i y_i p(c = i)). \tag{3}$$

In the special case of know object category label, the probability vector $p(c = i) = \delta(c^* = i)$ and Eqn. 3 simplifies to Eqn. 2. Also, note that it is valid to define $y_{wgt}(c) = \sum_i y_i p(c = i)$ because a weighted sum of axis-angle vectors is still an axis-angle vector. On the other hand, a weighted sum of rotation matrices is not guaranteed to be a rotation matrix. Taking the weighted sum of axis-angle vectors is analogous to projecting the corresponding rotation matrices to the tangent plane at identity, taking the weighted sum in the tangent plane and then projecting it back to the space of rotations.

Top-1 loss function: Instead of a weighted sum, if we choose the predicted object category label to be the one with the highest probability, *i.e.* $\hat{c} = \text{argmax}_i\, p(c = i)$, we get estimated pose $y_{top1}(c) = y_{\text{argmax}_i\, p(c=i)}$ leading to the pose loss

$$\mathcal{L}_p(y^*, y(c)) = \mathcal{L}_p(y^*, y_{\text{argmax}_i\, p(c=i)}). \tag{4}$$

3.3. Network training

We train the network in multiple steps. First, we fix the feature network to weights pre-trained for the ImageNet [9] image classification task. We then learn the category network and category-specific pose networks independent of each other. These are used to initialize the overall network which is then optimized at a lower learning rate with our new loss functions for the joint object category and pose estimation task. Like we mentioned earlier, we use the categorical cross-entropy loss for our category loss and the geodesic loss of Eqn. 1 for our pose loss. The same geodesic loss is also our evaluation metric and can be simplified using the Rodrigues' rotation formula, $R = I + \sin\theta[v]_\times + (1 - \cos\theta)[v]_\times^2$, to get viewpoint angle error (in degrees) between ground-truth rotation R^* and predicted rotation R,

$$\Delta(R, R^*) = \cos^{-1}\left[\frac{\text{trace}(R^T R^*) - 1}{2}\right]. \tag{5}$$

We use Adam optimizer [18] in all our experiments and our code was written in Keras [8] with TensorFlow backend [3].

4. Results and Discussion

We first present the data we use for our experimental evaluation in §4.1. Then, in §4.2, we present an analysis of a pre-trained ResNet=50 network for the task of pose estimation. In §4.3 we discuss a key assumption we make, that category dependent networks work better than category independent networks. We then present our results on the 3D pose estimation given object category label task in §4.4. Finally, in §4.5 we present our experiments on the joint object category and pose estimation task.

4.1. Datasets

For our experiments, we use the challenging Pascal3D+ dataset (release version 1.1) [38] which consists of images

of 12 common rigid object categories of interest: aeroplane (aero), bicycle (bike), boat, bottle, bus, car, chair, diningtable (dtable), motorbike (mbike), sofa, train and tvmonitor (tv). The dataset includes Pascal VOC 2012 images [2] and ImageNet images [9] annotated with 3D pose annotations that describe the position of the camera with respect to the object in terms of azimuth, elevation, camera-tilt, distance, image-translation and focal-length. We use the ImageNet-training+validation images as our training data, Pascal-training images as our validation data and Pascal-validation images as our testing data. Like we mentioned earlier, we concentrate on the problem of joint object category and 3D pose estimation assuming we have bounding boxes around objects returned by an oracle. We use images that contain un-occluded and un-truncated objects that have been annotated with ground-truth bounding boxes. There are a total of 20,843 images that satisfy this criteria across these 12 categories of interest, with the number of images across the train-val-test splits detailed in Table 1. We use the 3D pose jittering data augmentation technique proposed in [25] and the rendered images provided in [33] [1] to augment our training data.

Category	Train	Val	Test	Rendered
aeroplane	1765	242	244	198201
bicycle	794	108	112	199544
boat	1979	177	163	198949
bottle	1303	201	177	199641
bus	1024	149	144	198963
car	5287	294	262	194919
chair	967	161	180	196560
diningtable	737	26	17	195699
motorbike	634	119	127	199765
sofa	601	38	37	199888
train	1016	100	105	199718
tvmonitor	1195	167	191	199259
Total	17302	1782	1759	2381106

Table 1. Number of images in Pascal3D+ v1.1 [38] across various splits as well as rendered images provided by Su *et al*. [33].

4.2. Residual networks for 3D pose estimation

As mentioned earlier, one of our contributions is the use of the very popular residual networks designed by He *et al*. [14, 15] for the task of 3D pose estimation. Before we present our experimental results with residual networks, we would like to answer the question: Is there any merit in using residual networks at all? One way to answer the question is to see if the features/representations learned by these residual networks for the task of image classification generalize well to the task of pose estimation.

[1] https://shapenet.cs.stanford.edu/media/syn_images_cropped_bkg_overlaid.tar

Previous works like [10] (for AlexNet [20]), [25] (for VGG-M [7]) and [12] (for ResNet-101 [14]) have studied the applicability of features extracted from networks trained for the task of image classification, for the task of pose estimation. Image classification treats pose as a nuisance factor and the final output of networks trained for image classification are/should be pose invariant. However, [10], [25] and [12] have shown that the features extracted from intermediate layers retain information relevant to pose estimation.

Network	Features	d	Viewpoint error
ResNet-50	Stage-3	512	26.51
	Stage-4	1024	19.91
	Stage-5	2048	29.67
ResNet-101	Stage-4	1024	21.59

Table 2. Median viewpoint error averaged across object categories. These experiments use features extracted from different stages of pre-trained residual networks to train category-specific pose networks. Lower is better.

We extract features at different stages of the ResNet-50 architecture with pre-trained weights. We then learn 3-layer fully connected pose networks of size d-1000-500-3 using these features. As can be seen in Fig. 4 and Table 2, we find that features extracted at the end of Stage-4 are significantly better than the features extracted at the end of Stages-3 and 5. This is consistent with previous findings that show that (i) features become more specialized for the task they were trained for (image classification in this case) the deeper they are in the network, which explains why stage-4 features are better than stage-5, and (ii) deeper layers capture more complex information compared to simple edge detectors at the first few layers, which explains why stage-4 features are better than stage-3. We also perform the same experiment with features from another residual network, ResNet-101 Stage-4. We find that features extracted from the ResNet-50 network are better than those extracted from ResNet-101. More detailed results showing median viewpoint error for every object category are provided in Table 13.

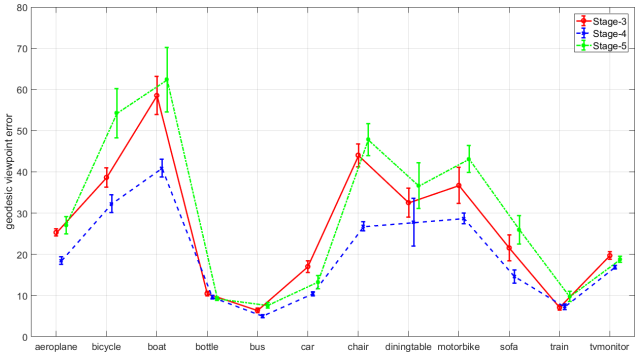

Figure 4. Median viewpoint error with features extracted from different stage of the pre-trained ResNet-50 network

4.3. Category-dependent pose networks

An implicit assumption we have made in our work is that our choice of category-dependent pose network architecture (also the choice of [35, 33, 25]) is better than the choice of a category-independent pose network (the choice of [10, 6]). This is based on the speculation that training category-specific pose networks learns features unique to that object category and helps improve pose estimation. In our architecture, some layers (the feature network) are shared between all object categories and some layers (the pose networks) are specific to each category. [10] discusses where the branching between category and pose estimation tasks should occur in their early branching (EBM) and late branching (LBM) models, but they do not discuss why they choose a category-independent pose network. We now validate our decision choice of category-specific pose networks empirically. For this experiment, we use the features extracted from ResNet-50 Stage-4. We then learn twelve 3-layer pose networks, one for each object category, of size 1024-1000-500-3. To compare with a category independent pose network, we use these same features to learn a single pose network of size 1024-12000-500-3. Note that the intermediate layer is of size 12000 to have roughly the same number of total parameters as the 12 independent pose networks. As can be seen in Table 3 (more detail in Table 14), solving for the pose in a per-category manner is better.

Method	Viewpoint error
Category-dependent	19.91
Category-independent	24.83

Table 3. Median viewpoint error averaged across the object categories for the category-dependent and independent pose networks

4.4. 3D pose estimation given object category

We now present our results on the task of 3D pose estimation given object category labels. In comparison to the experiments in Sec. 4.2 where the feature network was implicitly fixed, here, we train the overall network minus the category network *i.e*. we train the feature network and category-specific pose networks jointly with ground-truth object category labels. As we can see in Table 4 (more detailed results in Table 15), our method performs much better than 3D-pose-regression [25] which is also a regression based approach and slightly better than the pose-classification based approach of Viewpoints & Keypoints [35]. Our method is still behind Render-for-CNN [33] when averaged over all object categories but is better for 3 object categories. Ours-ResNet50-Stage4 refers to the network architecture where the feature model is ResNet-50 up to Stage-4 and similarly, Ours-ResNet50-Stage5 has the feature model of ResNet-50 up to Stage-5. They both have pose networks with intermediate layers of size 1000 and 500 respectively with a 3 dimensional output layer. Ours-ResNet50-Stage5+ has pose networks of intermediate sizes 2048 and 500. Ours-ResNet50-Stage5+ has the best performance of these three models and we attribute it to the increased size of its pose and feature networks.

Method	Viewpoint error
Viewpoints&Keypoints [35]	13.59
Render-for-CNN [33]	**11.67**
3D-pose-regression [25]	15.38
Ours-ResNet50-Stage4	14.13
Ours-ResNet50-Stage5	14.00
Ours-ResNet50-Stage5+	13.14
Ours-ResNet101-Stage4	14.01
Ours-ResNet152-Stage4	15.23

Table 4. Median viewpoint error for different methods averaged over 12 object categories. Lower is better. Best result in bold and second best in red.

As can be seen in Table 4, we perform better than [25] while solving a similar regression problem even though we use smaller feature network and pose networks (Table 5). This is because of three main reasons: (i) We used a residual network as the feature network, which allows for a more compact representation, (ii) We include rendered images during the finetuning step (unlike [25] which only used flipped images for finetuning), and (iii) We used manual balancing to include images of all object categories in every mini-batch. Manually balancing *i.e*. selecting a fixed number of images for each object category in every mini-batch is an alternative implementation of the recommendation in Ubernet [19] where they recommend asynchronous gradient updates to balance the variability of data across various tasks and datasets.

Method	Feature-network	Pose-networks
[25]	VGG-M (FC6)	12 x 4096-4096-500-3
Ours-stage4	ResNet-50 (Stage-4)	12 x 1024-1000-500-3
Ours-stage5	ResNet-50 (Stage-5)	12 x 2048-1000-500-3
Ours-stage5+	ResNet-50 (Stage-5)	12 x 2048-2048-500-3

Table 5. Feature and pose networks of different methods

As an ablative analysis, we run experiments to test the importance of including rendered images during the finetuning step and manually balancing the images in every mini-batch. For these experiments, we choose our feature network architecture to be ResNet-50 up to Stage-4 and our pose networks to be of size 1024-1000-500-3. As can be seen in Table 6, rendered images and manual balancing help reduce the viewpoint error. A more detailed table of these results can be found in Table 16.

We also test models with feature networks of ResNet-101 and ResNet-152 up to Stage-4. We expected these models to perform the best given the large number of parame-

include-rendered?	manual-balancing?	Viewpoint error
no	no	15.42
no	yes	15.40
yes	no	14.39
yes	yes	14.13

Table 6. Ablative analysis of including rendered images in finetuning and manually balancing object categories in every mini-batch

ters, however they did not. We speculate this is because: (i) the pre-trained feature network is more specialized for the task of image classification compared to ResNet-50, and (ii) larger networks were trained with smaller batch sizes and no manual-balancing to fit the network into a single GPU which is sub-optimal.

4.5. Joint object category and pose estimation

We now present the results of our experiments on the task of joint object category and pose estimation. As we mentioned earlier in Sec. 3.1, we choose our feature Network to be ResNet-50 up to Stage-4, our pose networks to be of size 1024-1000-500-3 and our category network to be ResNet-50 Stage-5. We train this network as described in Sec. 3.3 with the weighted loss and top-1 loss of Sec. 3.2. To evaluate our performance we report the object category estimation accuracy (cat-acc) and the median viewpoint error (pose-err), both averaged over all object categories.

Type of	Initial		Final	
pose output	cat-acc	pose-err	cat-acc	pose-err
Weighted	92.26	25.70	88.89	17.86
Top-1	92.26	25.76	87.60	17.51

Table 7. Object category estimation accuracy and median viewpoint error for the joint category and pose estimation task with different kinds of pose output. The "Initial" columns show performance with fixed feature network and independently trained category and pose networks. The "Final" columns show performance after finetuning the corresponding joint loss.

As can be seen in Table 7, there is a trade-off between the two tasks of object category estimation and 3D pose estimation. We lose some category estimation accuracy for significant improvement in pose estimation performance. This is to be expected as the tasks of object category and pose estimation are orthogonal to each other *i.e.* the estimated object category label ideally is invariant to object pose. The feature network tries to learn a representation that is suitable for both tasks and in doing so, we observer the trade-off.

Another way of initializing the network is to first train the feature network and pose networks end-to-end assuming known category labels like in Sec. 4.4 and then train the category network keeping the feature network fixed. We call this initialization "pose-init" and call the initialization strategy described in Sec. 3.3 "feature-init". As can be seen in Table 8, pose-init is better than feature-init for both types of pose outputs. This becomes clearer if we compare the number of trainable parameters in the three networks. The feature network has roughly 8M parameters, the pose networks have roughly 18M parameters and the category network has roughly 15M parameters. After we train the feature network + pose networks and fix them, the category network is able to compensate for pose specific features. We also tried different choices of λ, shown in Table 9.

Initialization	Weighted		Top-1	
	cat-acc	pose-err	cat-acc	pose-err
feature-init	88.89	17.86	87.60	17.51
pose-init	**89.44**	**16.07**	89.30	16.29

Table 8. Object category estimation accuracy and median viewpoint error for different initializations of the joint network. Higher category accuracy and lower pose error is better. Best results in bold.

λ	Weighted		Top-1	
	cat-acc	pose-err	cat-acc	pose-err
0.1	89.44	16.07	89.30	16.29
1	86.78	16.89	87.49	17.03

Table 9. Object category estimation accuracy and median viewpoint error for different choices of λ.

We also analyze performance when instead of the most-likely (Top-1) category label, the category network returns multiple labels (Top-2/3). To compute the pose error with multiple predicted category labels, we compute the viewpoint error with the pose output of every predicted category and take the minimum value. For *e.g.*, the pose error for the Top-3 category labels (c_1, c_2, c_3) using the notation of Sec. 3.2 is given by

$$\mathcal{L}_p(y^*, y(c_1, c_2, c_3)) = \min_{i=1..3} \mathcal{L}_p(y^*, y_{c_i}) \qquad (6)$$

As can be seen in Table 10, increasing the number of possible category labels leads to an increase in both category estimation accuracy and reduction in pose estimation error. However, it must also be noted that this reduction of pose error is very likely an artifact of the above metric because when we use an oracle for category estimation (GT), the viewpoint error is higher than Top-2/3 error. At the same time, improving category estimation accuracy (comparing Top-1 and GT, $89.30 \rightarrow 100$) leads to better performance in pose estimation ($16.29 \rightarrow 15.28$).

Top-1		Top-2		Top-3		GT	
acc	err	acc	err	acc	err	acc	err
89.30	16.29	94.55	14.08	95.95	13.15	100	15.28

Table 10. Object category estimation accuracy and median viewpoint error when Top-1/2/3 predicted labels are returned by the category network. More details in text.

We also compare our performance with that of Elhoseiny *et al*. [10], the current state-of-the-art on the joint object category and pose estimation task on the Pascal3D+ dataset. They report performance of azimuth angle estimation using the following metrics: (i) P%($< 22.5°/45°$): percentage of images that have pose error less than $22.5°/45°$ and (ii) AAAI: $1 - [\min(|err|, 2\pi - |err|)/\pi]$. We evaluate our models using these metrics for both azimuth error $|az - az^*|$ and 3D pose error $\Delta(R, R^*)$. For azimuth estimation, we predict 3D rotation matrix and then retrieve the azimuth angle. As can be seen in Table 11, we perform better than [10] in both object category and pose estimation accuracy.

Method	cat-acc	P% (< 22.5)	P% (< 45)	AAAI
EBM(800) [10]	83.79	51.89	60.74	75.39
Ours-Joint(wgt)	**89.44**	62.35	**75.79**	**80.09**
Ours-Joint(top1)	89.30	**62.68**	75.55	79.65
Ours-Joint(wgt)	89.44	64.09	78.35	81.88
Ours-Joint(top1)	89.30	63.95	78.32	81.84

Table 11. Comparing our joint networks with weighted (wgt) and top-1 (top1) pose output with the EBM(800) model (best results) of [10] with Train:Pascal+ImageNet and Test:Pascal. Rows 2-3 show performance for azimuth estimation and Rows 4-5 for 3D pose estimation. Higher is better and best results in bold.

4.6. Experiments on the RGBD dataset

The RGBD dataset [21] consists of images of 300 object instances from 51 object categories. These objects were placed on a turn-table and imaged at different azimuth angles for three elevations of $30°$, $45°$ and $60°$. Of these 51 object categories, we selected a subset of 24 categories that do not have cylindrical symmetry (Appendix. A). We train on the images at elevations of $30°$ and $60°$ and test on the images at elevation $45°$. We augment the training data by jittering the original images using in-plane rotations from $-5°$ to $5°$ in steps of $0.1°$. Our network architecture for this experiment is similar to our earlier network: ResNet-50 up to Stage-4 as the feature network and ResNet-50 Stage-5 for the category network. However, due to limited data we use a simpler 2-layer pose network. The first fully connected (FC) layer is shared between all object categories and is of size 100. The second layer is a fully connected layer of size 1 and activation $180 \times \tanh$ for every object category. We also include batch normalization layers (BN) and rectified linear units (ReLU) between the intermediate layers. In shorthand notation, our pose network is : Input(1024) - BN - FC(100) - BN - ReLU - {FC(1) - $180 \times \tanh$}$\times$24. We train the pose networks with mean absolute error loss and Adam optimizer. We train the category network with categorical cross-entropy loss and Adam optimizer. Note that in this experiment, we do not finetune the overall network *i.e.* the feature network is fixed to pre-trained weights and only the pose networks and category network are trained.

Method	cat-acc	P%(< 22.5)	P%(< 45)	AAAI
EBM(800)* [10]	97.14	**66.13**	77.02	78.83
Ours-Joint(wgt)	**99.43**	54.56	79.38	84.14
Ours-Joint(top1)	**99.43**	54.63	**79.44**	**84.16**

Table 12. Comparing our joint networks with weighted (wgt) and top-1 (top1) pose output with the EBM(800) model (best results) of [10] on the RGBD dataset. *Note that [10] uses 34 out of 51 object categories in their evaluation and emails to the authors asking which subset of categories were chosen were unanswered.

5. Conclusion

We have designed a residual network based architecture for the task of joint object category and 3D pose estimation. We have developed two new formulations and loss functions for category-specific pose networks when category label is not known: Weighted pose and Top-1 pose. We have shown state-of-the-art performance on the Pascal3D+ and RGBD datasets. We have analyzed our various decision choices, like : residual networks for 3D pose estimation, category-specific pose networks and manual-balancing, and shown empirical evidence for the same. We have also analyzed different initializations for the joint task and showed that "pose-init" is better than "feature-init". We have also tested our proposed networks on the 3D pose estimation with known object category task and shown competitive performance on the Pascal3D+ dataset.

Acknowledgements This research was supported by NSF grant 1527340. The research project was conducted using computational resources at the Maryland Advanced Research Computing Center (MARCC). This work also used the Extreme Science and Engineering Discovery Environment (XSEDE) [34], which is supported by National Science Foundation grant number OCI-1053575. Specifically, it used the Bridges system [28], which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

A. RGBD object categories

Chosen: banana, calculator, camera, cap, cell_phone, cereal_box, coffee_mug, comb, dry_battery, flashlight, food_bag, food_box, instant_noodles, keyboard, light_bulb, marker, notebook, pitcher, pliers, rubber_eraser, scissors, stapler, toothbrush, toothpaste

Discarded: apple, ball, bell_pepper, binder, bowl, food_can, food_cup, food_jar, garlic, glue_stick, greens, hand_towel, kleenex, lemon, lime, mushroom, onion, orange, peach, pear, plate, potato, shampoo, soda_can, sponge, tomato, water_bottle

Features	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	mean
ResNe-50 Stage-3	25.32	38.65	58.55	10.47	6.42	17.02	44.06	32.56	36.72	21.58	7.12	19.70	26.51
ResNet-50 Stage-4	18.44	32.29	40.91	9.65	5.01	10.45	26.79	27.78	28.70	14.64	7.29	16.92	19.91
ResNet-50 Stage-5	27.07	54.27	62.40	9.24	7.59	13.26	47.84	36.67	43.11	25.99	9.78	18.79	29.67
ResNet-101 Stage-4	19.74	36.82	49.21	9.13	5.09	11.44	26.37	29.80	30.37	15.44	8.36	17.36	21.59
VGGM FC6 [25]	15.56	22.98	40.29	9.09	4.92	8.06	22.21	34.88	22.13	14.09	7.88	16.67	18.23

Table 13. Median viewpoint error after learning pose networks using features extracted from pre-trained networks. Pose networks are of size 512/1024/2048-1000-500-3 for ResNet-50 Stages-3/4/5 respectively. ResNet-101 Stage-4 and VGGM FC6 are of size 1024-1000-500-3 and 4096-4096-500-3 respectively. Lower is better.

Features	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	mean
Category-dependent	18.44	32.29	40.91	9.65	5.01	10.45	26.79	27.78	28.70	14.64	7.29	16.92	19.91
Category-independent	23.93	40.92	54.71	12.07	5.76	11.63	41.80	26.94	31.01	20.34	9.95	18.91	24.83

Table 14. Median viewpoint error for category dependent and independent pose networks across different object categories. Lower is better.

Methods	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	Mean
Viepoints&Keypoints [35]	13.80	17.70	**21.30**	12.90	5.80	9.10	14.80	15.20	**14.70**	13.70	8.70	15.40	13.59
Render-for-CNN [33]	15.40	**14.80**	25.60	9.30	3.60	**6.00**	**9.70**	**10.80**	16.70	**9.50**	**6.10**	**12.60**	**11.67**
3D-pose-regression [25]	13.97	21.07	35.52	**8.99**	4.08	7.56	21.18	17.74	17.87	12.70	8.22	15.68	15.38
Ours-ResNet50-Stage4	**12.67**	20.07	31.36	9.55	3.23	6.70	16.72	18.92	17.00	12.69	6.21	14.39	14.13
Ours-ResNet50-Stage5	13.52	19.79	28.12	9.54	3.24	7.17	16.47	21.48	16.14	11.79	6.50	14.19	14.00
Ours-ResNet50-Stage5+	14.24	18.71	27.17	9.54	**3.01**	6.91	15.75	14.40	16.35	10.74	6.59	14.25	13.14
Ours-ResNet101-Stage4	13.95	20.55	29.15	10.01	3.30	6.04	15.50	19.79	17.13	11.63	6.53	14.52	14.01
Ours-ResNet152-Stage4	14.14	20.85	32.92	9.40	3.89	7.22	17.42	23.73	18.94	13.07	6.17	14.96	15.23

Table 15. Median viewpoint error for the 3D pose estimation task with known object category labels across different object categories. Descriptions of different methods are provided in Sec. 4.4. Best results are in bold and second best in red. Lower is better.

Expt.	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	Mean
(no, no)	12.75	23.82	32.01	9.40	3.36	7.36	25.46	**15.86**	18.94	14.41	6.79	14.82	15.42
(no, yes)	12.69	22.21	33.94	**8.95**	3.29	7.51	24.68	19.24	18.19	12.86	6.30	14.93	15.40
(yes, no)	**12.07**	21.55	**29.23**	9.29	3.65	**6.38**	18.10	18.82	18.58	13.65	6.72	14.61	14.39
(yes, yes)	12.67	**20.07**	31.36	9.55	**3.23**	6.70	**16.72**	18.92	**17.00**	**12.69**	**6.21**	**14.39**	**14.13**

Table 16. Detailed ablative analysis of Table 6. Each experiment corresponds to different choices of (include-rendered?, manual-balancing?). Best results in bold and second best in red. Lower is better.

Expt.	Type	Metric	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	mean
Feature-init Initial	Both	cat-acc	**96.84**	90.76	**97.14**	**98.46**	95.45	92.06	**89.92**	**71.43**	**97.10**	84.62	94.69	**98.65**	**92.26**
	Weighted	pose-err	25.46	39.04	47.61	13.45	9.74	15.18	34.61	38.17	37.50	21.47	9.58	16.63	25.70
	Top-1	pose-err	25.46	38.87	49.52	13.45	9.36	14.51	35.83	38.17	37.23	21.47	8.87	16.69	25.79
Feature-init Final	Weighted	cat-acc	91.58	89.36	96.33	96.01	95.02	81.27	72.72	**71.43**	93.96	**87.18**	**95.58**	96.25	88.89
		pose-err	13.90	24.14	35.42	9.64	4.11	8.80	33.53	24.06	19.49	17.70	7.56	15.94	17.86
	Top-1	cat-acc	89.71	88.52	97.41	88.03	93.94	79.37	69.62	73.02	94.69	**87.18**	94.69	95.05	87.60
		pose-err	13.70	23.92	**32.36**	10.57	3.82	8.02	32.53	26.29	17.79	18.03	7.65	15.40	17.51
Pose-init $\lambda = 0.1$	Weighted	cat-acc	94.39	88.80	95.10	97.68	95.67	93.97	82.80	66.67	93.48	77.78	93.51	93.39	89.44
		pose-err	13.60	**21.26**	34.79	**9.05**	3.43	7.78	26.62	20.82	17.65	**15.76**	6.89	15.19	**16.07**
	Top-1	cat-acc	96.61	87.39	94.01	96.14	**95.89**	**95.34**	83.60	61.90	95.89	77.78	94.40	92.64	89.30
		pose-err	13.39	22.20	33.50	9.21	**3.34**	**7.66**	**26.16**	24.00	17.79	16.49	**6.61**	**15.18**	16.29
Pose-init $\lambda = 1$	Weighted	cat-acc	92.28	**91.04**	96.19	91.25	94.37	81.90	80.11	69.84	85.27	75.21	91.15	92.79	86.78
		pose-err	14.48	23.77	38.40	9.50	3.43	8.43	28.39	**18.62**	**17.55**	17.43	7.28	15.38	16.89
	Top-1	cat-acc	91.58	89.92	93.06	87.64	94.81	85.08	73.39	**71.43**	90.58	**87.18**	94.69	93.69	87.75
		pose-err	**13.35**	22.82	36.76	10.17	3.44	8.48	30.22	19.24	18.56	15.81	6.70	15.36	16.74

Table 17. Object category estimation accuracy (cat-acc) and pose viewpoint error (pose-err) for experiments with joint networks. Detailed descriptions are provided in Sec. 4.5. Higher is better for cat-acc and lower is better for pose-err. Best results in bold.

References

[1] A coarse-to-fine model for 3D pose estimation and sub-category recognition. 2

[2] The PASCAL Object Recognition Database Collection. http://www.pascal-network.org/challenges/VOC/databases.html. 4

[3] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 3

[4] M. Aubry, D. Maturana, A. A. Efros, B. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2

[5] A. Bakry, T. El-Gaaly, M. Elhoseiny, and A. Elgammal. Joint object recognition and pose estimation using a nonlinear view-invariant latent generative model. In *IEEE Winter Applications of Computer Vision Conference*, 2016. 2

[6] M. Braun, Q. Rao, Y. Wang, and F. Flohr. Pose-RCNN: Joint object detection and pose estimation using 3D object proposals. In *International Conference on Intelligent Transportation Systems*, 2016. 2, 5

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. 2014. 2, 4

[8] F. Chollet. Keras. https://github.com/fchollet/keras, 2015. 3

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 3, 4

[10] M. Elhoseiny, T. El-Gaaly, A. Bakry, and A. Elgammal. A comparative analysis and study of multiview CNN models for joint object categorization and pose estimation. In *International Conference on Machine learning*, 2016. 2, 4, 5, 7

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 1

[12] K. Hara, R. Vemulapalli, and R. Chellappa. Designing deep convolutional neural networks for continuous object orientation estimation. *CoRR*, abs/1702.01499, 2017. 2, 4

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision*, 2017. 1

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2, 4

[15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. 2, 4

[16] M. Hejrati and D. Ramanan. Analysis by synthesis: 3D object recognition by object reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2

[17] R. Juranek, A. Herout, M. Dubska, and P. Zemcik. Real-time pose estimation piggybacked on object detection. In *IEEE International Conference on Computer Vision*, 2015. 2

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014. 3

[19] I. Kokkinos. Ubernet: Training a 'universal' convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 5

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, 2012. 4

[21] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *IEEE International Conference on Robotics and Automation*, 2011. 7

[22] W. Li, Y. Luo, P. Wang, Z. Qin, H. Zhou, and H. Qiao. Recent advances on application of deep learning for recovering object pose. In *International Conference on Robotics and Biomimetics*, 2016. 2

[23] R. Lopez-Sastre, T. Tuytelaars, and S. Savarese. Deformable part models revisited : A performance evaluation for object category pose estimation. In *ICCV2011 Workshops*, 2011. 2

[24] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3D Vision: From Images to Geometric Models*. Springer Verlag, 2003. 3

[25] S. Mahendran, H. Ali, and R. Vidal. 3D pose regression using convolutional neural networks. In *IEEE International Conference on Computer Vision*, Oct 2017. 1, 2, 4, 5, 8

[26] F. Massa, M. Aubry, and R. Marlet. Convolutional neural networks for joint object detection and pose estimation: A comparative study. *CoRR*, abs/1412.7190, 2014. 2

[27] F. Massa, R. Marlet, and M. Aubry. Crafting a multi-task CNN for viewpoint estimation. *CoRR*, abs/1609.03894, 2016. 2

[28] N. A. Nystrom, M. J. Levine, R. Z. Roskies, and J. R. Scott. Bridges: A uniquely flexible hpc resource for new communities and data analytics. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, XSEDE '15, pages 30:1–30:8, New York, NY, USA, 2015. ACM. 7

[29] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-DoF object pose from semantic keypoints. In *IEEE International Conference on Robotics and Automation*, 2017. 2

[30] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3D geometry to deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 2

[31] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 1

[32] S. Savarese and L. Fei-Fei. 3D generic object categorization, localization and pose estimation. In *IEEE International Conference on Computer Vision*, 2007. 2

[33] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *IEEE International Conference on Computer Vision*, December 2015. 1, 2, 4, 5, 8

[34] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. XSEDE: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74, 2014. 7

[35] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1, 2, 5, 8

[36] Y. Wang, S. Li, M. Jia, and W. Liang. Viewpoint estimation for objects with convolutional neural network trained on synthetic images. In *Advances in Multimedia Information Processing - PCM 2016*, 2016. 2

[37] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman. Single Image 3D Interpreter Network. In *European Conference on Computer Vision*, pages 365–382, 2016. 1, 2

[38] R. M. Yu Xiang and S. Savarese. Beyond PASCAL: A benchmark for 3D object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, 2014. 1, 3, 4