Target Driven Instance Detection

Phil Ammirato
UNC-Chapel Hill

Cheng-Yang Fu
UNC-Chapel Hill

Mykhailo Shvets
UNC-Chapel Hill

Jana Košecká
George Mason University

Alexander C. Berg
UNC-Chapel Hill

July 17, 2018

Abstract: While state-of-the-art general object detectors are getting better and better, there are not many systems specifically designed to take advantage of the instance detection problem. For many applications, such as household robotics, a system may need to recognize a few very specific instances at a time. Speed can be critical in these applications, as can the need to recognize previously unseen instances. We introduce a Target Driven Instance Detector(TDID), which modifies existing general object detectors for the instance recognition setting. TDID not only improves performance on instances seen during training, with a fast runtime, but is also able to generalize to detect novel instances.

Keywords: Vision, Object Detection

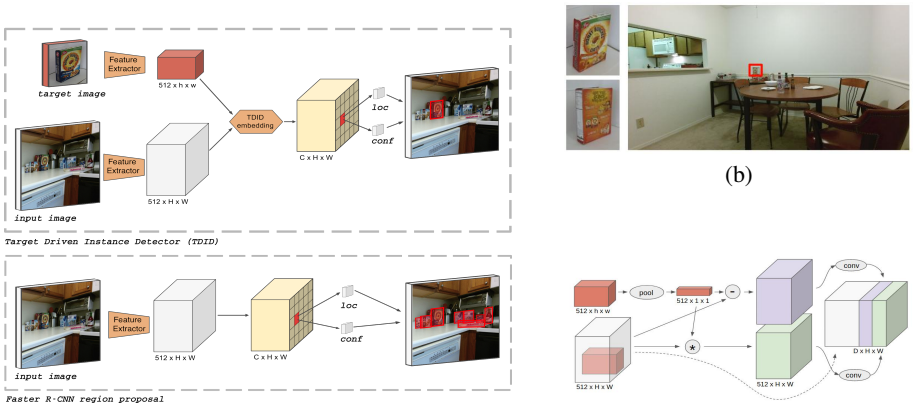
1 Introduction

Object detection works! Alas, this is not always true, and the specific version of object detection matters. There have been massive improvements in the accuracy of *category-level* object detectors based on deep learning [1, 2]. These require many labeled training examples of bounding boxes for each category (e.g. mug) in question, use carefully crafted data augmentation approaches to fully leverage that training data, and can take days or longer to train. This leaves out an important type of object detection problem where the goal is to detect a precise instance of an object category (e.g. my mug instead of a mug). This setting applies to real world tasks including fetch and deliver in household environments, and robotic manipulation in industrial environments, where the objects in question are often specific instances and not general categories. The instance task does not have the large intra-class variation of category-level detection, and sometimes only a small number of training examples for each instance is available.

How can the progress on category-level object detection be harnessed and applied to instance detection, taking advantage of the specificity of instances and overcoming the challenge of small numbers of training examples? One approach is to take a small number of example images and artificially create a large number of detection training examples by artificially composing those examples into scenes [3, 4]. This still treats instance detection as a category detection problem, but expands a small number of clean images of an object instance into enough samples to train current category detectors. Another possible approach reduces the training necessary for new targets by preconditioning a network to be robust to varying views of objects. Recent work by Held et al. [5] has shown good accuracy with such an approach, and that a deep-learning-based method for learning a classifier from single examples can be more accurate than a wide range of previous template matching approaches.

This paper presents a new approach that goes further than the two above by learning a detector that directly takes advantage of the uniqueness of instances, and that does not need to be retrained or fine tuned in order to detect a new target object. This is done by learning an embedding that compares learned features of the target to learned features at each location in a scene image, and integrating this into a state-of-the-art detection framework.

Submitted to the 2nd Conference on Robot Learning (CoRL 2018). Do not distribute.



(a) (b) (c)

Figure 1: **(a)** Bottom box is Faster R-CNN's RPN. Top box is our TDID model. We enrich the feature representation with a joint embedding. TDID extracts features from scene and target image, combines those with a novel TDID embedding module, and applies the detection prediction head. **(b)** Example of target images, and an input "scene image" that contains the target object in a different pose, partially occluded, at small scale. The object's bounding box (in red) for reference. **(c)** TDID embedding: given a pair of scene (gray) and target (red) features, makes a joint tensor embedding. Target features are pooled and then depth-wise correlated with (*) and subtracted from (-) scene features. In final model scene features (IMG in Table 1, dotted line and white box here) are not used.

The Target Driven Instance Detection problem is formulated as follows: given an input scene image S and a number, T, of images of a target object, output a bounding box around the target object in S, or no box if the object is not present. See Figure 1b for an example with $T = 2$ target images with the correct output shown. The Target Driven Instance Detector (TDID) is constructed by augmenting the internal representation of an object detector by embedding the scene image's features together with the target features. This constructs *joint embedding* features for a scene-target pair, which are specialized for target instance detection and implicitly compare the target to each location in the image. The final prediction layers can then use this information as part of end-to-end training of the detection framework.

The datasets [6, 7, 8] we use to evaluate the Target Driven Instance Detector display a set of object instances in everyday home environments and exhibit real-world confounding factors such as large scale variation, small instances, clutter and occlusion. An important aspect of the Active Vision Dataset [6] (used for test in many experiments) is that it was collected to sample views of household rooms from every position where a robot could navigate. As a result, many objects are quite small in some views, perhaps when seen across a room, and are partially occluded in many views.

The objects used in our experiments come from the BigBird and RGB-D Object datasets [9, 8], and we note that part of the methods success stems from seeing similar objects in training. This is the same in previous work to which we compare, and is reasonable to expect in real-world mobile manipulation applications, but it is important to make this clear.

We summarize our contributions as follows: *(1)* A novel *Target Driven Instance Detector* (TDID) model that easily transforms the current state-of-the-art general object detectors into instance detectors, as depicted in Figure 1a. *(2)* Strong performance improvement on multiple challenging instance detection scenarios, as demonstrated with experiments. We significantly improve in accuracy over previous results using general object detectors, while maintaining competitive inference speed. *(3)* We compare TDID to previous work on one-shot training for instance classification, and show better accuracy. By proxy, this shows that TDID provides better accuracy than a range of previous template matching techniques. *(4)* The ability to generalize detection to unseen instances on challenging datasets without any additional training or fine-tuning.

2 Related work

We discuss recent history of detectors up to the current state-of-the-art deep-learning-based object category detectors, then move to instance recognition and briefly touch on some related vision problems including tracking and navigation.

Traditional methods for object detection in cluttered scenes follow the sliding window based pipelines where efficient methods for feature computation and classifier evaluation were developed such as DPM [10]. Examples of using these models in the table top setting include [11, 12]. Object detection and recognition systems that deal with textured household objects such as Collet et al. [13] and Tang et al. [14] take advantage of the discriminative nature of local descriptors. A disadvantage of these local descriptors is that they usually perform poorly in the presence of non-textured objects. Some of these issues were tackled by [15] which used template based methods to deal with such texture-less objects. Hand engineered features typically work well in table top settings that contain a relatively small number of objects at relatively large scale [16]. The authors in [17] introduce an effective approach to feature learning for simultaneous categorization and pose estimation for single objects on uniform backgrounds.

General Object Detector State-of-the-art object-category detectors have been improved significantly over the last few years in both accuracy and speed. These detectors rely on a backbone architecture, such as VGG [18] or ResNet [19], to extract features from the image, and then add a detection module on top of these features. Two-stage detectors, such as Faster R-CNN [1], and R-FCN [20], rely on an initial region proposal followed by a classification and location regression of the proposed regions. Recent single-stage detectors: YOLO [21], YOLOv2 [22], and SSD [2] skip the feature pooling stage and show that fast inference speed can be achieved. Recent work has added top-down connections [23, 24, 25], which can borrow rich semantic information from deeper layers and show improvement in accuracy for small objects. though usually at reduced speed.

Instance Recognition Compared to object-category recognition, the specific instance recognition setting has less intra-class variation and, in practice, is often allowed only a limited number of training examples. Much work has been done using hand-crafted features and template matching to identify object instances even since somewhat recent seminal work [26, 27]. More recently hand-crafted features have been replaced with learned ones in many recognition tasks [28]. The instance recognition dataset BigBIRD [9], which provides dense, individual scans of over 100 object instances on a turntable has enabled more research on instance recognition. [5] shows that pre-training on BigBIRD improves robustness to pose and improves classification performance over hand-crafted and template matching methods, even if only one image per instance is provided for training.

Instance Detection The recent release of larger scale instance detection datasets like the Active Vision Dataset(AVD) [6] and GMU Kitchens [7], has enabled more work using deep learning for instance detection. The GMU Kitchen Dataset has 6,728 images across 9 scenes, and the initial release of AVD has 17,556 images across 9 scenes. Both datasets feature instances very similar to those in BigBIRD [9], with GMU featuring 11 such instances and AVD 30. [3, 4] attack the problem of limited training examples by synthesizing new examples with different background images. In both of these works general object category detectors such as SSD [2] or Faster R-CNN [1] are still used to solve the instance detection problem.

Navigation Zhu et al. [29] address a related problem, exploring an environment to reach a target position. They also input both a target image (of the desired view) and an image from the current position, and learn an embedding to aid in navigation. It is not straightforward to adapt their method to the instance detection problem, however, as they aim to move so that the image at the current position matches the target exactly. The embedding is not designed to localize objects, which is necessary for detection. Furthermore, the network requires scene-specific layers, while most object detectors are expected to generalize to unseen environments.

Tracking Given an initial bounding box of an object, the tracking task is to localize the same object appearing in each subsequent video frame. Correlation is frequently used for estimating similarity of patches between frames. [30] applies correlation on histogram features. [31] kernelizes the correlation filter to improve accuracy. [32] targets scale issues by running the correlation filter on spatial pyramid of features. Recent deep learning methods, such as [33], uses a Siamese network to measure the similarity in tracking. [34] uses a correlation filter to transform the Siamese network to be fully convolutional. [35] combines the features of crops from previous and current frames to

regress the location directly. [36] interprets the correlation filter learner as a differentiable layer and enables learning deep features that are tightly coupled to the correlation filter. Strong priors on the object and background exist in tracking, namely that neither changes much frame to frame. These priors include scale, location, illumination, orientation and viewpoint. Our instance detection setting requires robustness to larger changes between target and scene.

3 Method

3.1 Problem Formulation

Instance detection requires a system to recognize and localize specific objects in novel images. Usually these images contain many objects, some of which are instances to be recognized. We will refer to these images as *scene images*. Most object detectors work by training on a set of scene images and ground truth bounding boxes of objects, and then test on novel scene images containing the same types of objects. General object detectors attempt to find all object instances in a scene image at once.

Our Target Driven Instance Detector (TDID), takes as input not only a scene image, but also one or more *target images*. These target images contain only the instance of interest, see Figure 1b for an example. TDID attempts to detect only this target instance in the scene image.

3.2 Network Architecture

TDID takes the Region Proposal Network (RPN), the first stage of Faster-RCNN, and adds a target/scene joint embedding. Figure 1a compares our architecture with that of the RPN. With this joint embedding we are able to outperform other detectors even without the second stage of the traditional Faster-RCNN pipeline. This results in an architecture that computes detection outputs in one shot, with speed close to other one-shot detectors, while achieving better accuracy on various instance detection tasks than both one and two stage detectors.

The high-level view of our architecture is as follows: Extract features from the target and scene images using some shared feature extraction network, such as VGG-16[18]. Next, pass both target and scene image feature maps through our joint embedding. Finally, a set of convolutions predict class scores and bounding box regression parameters for a set of default anchors boxes (see Figure 1b) over the embedding feature map. In TDID there are only two classes: target object or background.

Joint Embedding We construct a joint embedding, see Figure 1c, of all input images that can then be further processed for detection. The joint embedding combines feature correlation and differencing between the target image(s) and the scene image. The operations and features in the embedding are described below and Table 1 shows ablation results as different feature combinations are considered.

Cross Correlation is widely used in traditional methods with hand-crafted features for similarity matching. We started building the joint embedding by applying the cross correlation of target features with the scene features, generating a heatmap with only one channel dimension. This method generates a strong signal for predicting target presence/absence in each spatial location, but drops rich information from the feature channels. *Depthwise-separable correlation* applies correlation at each channel independently. This not only preserves more information for the subsequent instance localization but also yields high computational efficiency [37, 38]. We use depthwise-separable correlation in our joint embedding, represented as CC in the ablation study, Table 1, and the green box in Figure 1c.

Feature differencing is another way to measure similarity. Intuitively, a network attempting to learn a similarity between image features may do something like learn to subtract them. Instead of adding extra complexity to our framework by learning a similarity, we compute the difference directly and feed it as a signal to our joint embedding. We first apply global max pooling on the target features to bring them to 1×1 spatial resolution. Then we subtract this vector from each spatial location of the scene features. This feature is represented as $DIFF$ in the ablation study, Table 1, and the purple box in Figure 1c.

Scene Image Features The features of the scene image from the feature extractor may also provide useful information for object detection. In the original RPN of Faster-RCNN, these are the features

Features Used	extra small	small	medium	large	mAP
IMG	1.9	7.7	5.1	5.3	2.2
CC	23.8	58.5	44.0	50.7	27.7
DIFF	48.0	74.6	72.3	73.2	52.6
IMG+CC	28.0	54.5	51.4	54.8	31.9
IMG+DIFF	46.2	79.2	72.5	71.3	50.9
CC+DIFF	**50.3**	78.2	**75.1**	**78.2**	**55.8**
IMG + CC + DIFF	48.4	**83.0**	73.8	77.1	53.3

Table 1: Ablation study of features to be included in TDID embedding. IMG == scene image features, CC == cross-correlation, and DIFF == difference. Train/Test on AVD split 2. Sizes defined as in [6].

that are used to predict bounding boxes and potential objects. This feature is represented as IMG in the ablation study, Table 1, and the white box in Figure 1c.

Ablation Study We run an ablation study to show how using different combinations of features in our joint embedding affects detection performance. Results are reported for the instance detection task on split 2 of the AVD dataset. We choose split 2 as the results for split 1 in [6] were lower than the other splits, and we wanted to see results on a more typical scenario. As expected, using just scene image features, IMG, fails as there is no information about the target instance. Surprisingly, using just $DIFF$ features provides a strong signal resulting in high detection performance. The addition of CC features provides a small boost in performance here, and also proved to be useful in later experiments so it is included in our final model. IMG features do not provided much new information from $DIFF$ and CC, while adding extra complexity and parameters to the network. We do not use these features in the embedding in our final model.

Final Embedding Our final joint embedding first pools the target features to be $N \times 1 \times 1$ where N is the number of channels in the feature map outputted by the feature extractor. This pooled target feature vector is then both cross-correlated with, and subtracted from, every location in the scene image feature map. These features, CC and $DIFF$, are then each passed through their own 3×3 convolution to reduce the feature dimension to $\frac{N}{2}$. The IMG features, represented by the dotted skip connection and white box in Figure 1c, are not used in the final model. The CC and $DIFF$ features are then concatenated and passed through a final 3×3 convolution before being sent to the classification and regression filters.

Figure 1a shows the model for one target image and one scene image. In general, many target images may be used, providing more views of the target instance. Each target image will generate its own set of CC and $DIFF$ features, which will all be concatenated before going through their respective 3×3 convolutions.

Training To construct the training loss, we follow the region proposal settings in Faster R-CNN. For box localization regression we use the Smooth L_1 error. Each anchor box is matched to the ground-truth target object box if its intersection-over-union (IoU) with the ground truth is over 0.6 and to background if its IoU is lower than 0.3. Since we are only looking for one object at a time, there are only two possible classes for each anchor box: target or background.

Inference During inference, we run one input/target pair at a time. In each case we select at most 5 detections after non-maximum suppression with 0.7 IoU threshold. We use IoU=0.5 as the matching criteria and modify the COCO evaluation parameters[1] for our experiments to report accurate mean Average Precision (mAP) results.

4 Experiments

We evaluate our method on three tasks: object instance detection, one-shot instance classification, and few-shot object instance detection. For all TDID models we use Pytorch [39], CUDA 8.0, and cuDNN v6.

[1] https://github.com/cocodataset/cocoapi

Method	Backbone	image size	speed
SSD[2]	VGG16	512x512	19fps
Faster-RCNN[1]	VGG16	600x1000	5fps
TDID	VGG16	960x540	12fps
TDID	VGG16	720x405	19fps

Table 2: Speed of various object detectors. Faster-RCNN[1] and SSD[2] speeds are reported in their respective papers.

# of Instances	1	2	5	10
TDID (960x540)	12fps	10fps	6fps	4fps
TDID (720x405)	19fps	16fps	10fps	6fps

Table 3: How the inference speed of TDID changes when detecting multiple instances in a single scene image, on a TITAN X GPU.

4.1 Object Instance Detection

For all of our object instance detection experiments, we report the same mAP as regular object detection. Since our system only considers one object at a time, to calculate mAP fairly we test all pairs of target object and scene image on our system. For example for the AVD dataset there are 30 instances. So for each image in the test set we run our network (or part of it, see below) 30 times, once for every instance. A general object detector runs once per image, and outputs boxes for every class. This seems like a big disadvantage for our system, since it is cumbersome to run the network for every single instance. In fact, this is where our system gains its advantage. In many applications, the system will only be looking for one, or very few, object(s) at a time. Our network is able to take advantage of this to greatly increase performance.

It should also be noted that **we do not need to run our entire system multiple times for multiple targets in one scene image**. Once the model is trained, all target features through the backbone feature extractor can be pre-computed and stored. Then features are extracted for each scene image once, and we only run the joint embedding and detection head of the network for each target. See Table 3 for a study of how inference time changes as more instances are detected in a single scene image.

The speed/accuracy trade-off of object detectors has become of great interest in recent years[40] as general object detectors get better and faster. Table 2 compares the speed of TDID with the reported speeds of Faster-RCNN and SSD. TDID is a lightweight detector and can achieve speeds much faster than Faster-RCNN, approaching that of SSD while operating on higher resolution images. As we show in our experiments, while improving or maintaining speed, TDID can also improve instance detection performance over these general object detectors. It is well-known that use of various feature extraction backbone networks can influence detection performance. To keep all comparisons fair to reported results, we use VGG-16 as the backbone network in our experiments.

4.1.1 Active Vision Dataset

We first evaluate our system on a challenging object instance detection dataset, AVD [6]. We use two target images (provided on the dataset website) for each instance, picking views to maximize how much of the object is seen. See Figure 1b for an example of two target images. We choose two target images because in general it may be impossible to recognize an instance from the back if only the front view is provided.

We report results for all three train/test splits reported in [6]. We resize all images to $960x540$ for training, and supply more training details in the appendix. Table 4 shows that our method outperforms SSD[2, 6] on this task consistently, over 14 mAP on each split on all boxes (boxes $> 50 \times 30$).

To produce a TDID system that runs at the same frame rate as SSD, we resize all images during testing to $720x405$. We test the same model that was trained on the $960x540$ images, and show results in the TDID(720x405) row in Table 4. We can see TDID still outperforms SSD by an average of over 5 mAp on all objects, and over 20 mAP on larger objects. We expect training a model at this resolution could result in even greater accuracy gains, while maintaining speed.

Method	Backbone	Box Size	Split 1	Split 2	Split 3
SSD[6]	VGG16		39	55	53
TDID(720x405)	VGG16	$> 100 \times 50$	65.6	71.6	72.1
TDID(960x540)	VGG16		**70.3**	**75.4**	**72.7**
SSD[6]	VGG16		26	41	42
TDID(720x405)	VGG16	$> 50 \times 30$	35.8	42.7	48.2
TDID(960x540)	VGG16		**48.9**	**55.8**	**56.5**

Table 4: Instance detection results (mAP) on the AVD dataset.

Train set	Method	coca cola	honey bunches	hunt's sauce	mahatma rice	nature v2	red bull	mAP
Real Images	Faster RCNN	**57.7**	34.4	48.0	39.9	24.6	46.6	41.9
Real Images	TDID	57.4	**34.5**	**73.8**	**43.3**	**32.1**	**57.0**	**49.7**
Real + Synthetic*	Faster RCNN	**69.9**	44.2	51.0	41.8	48.7	50.9	51.1
Real + Synthetic	TDID	69.1	**46.9**	**69.7**	**43.0**	**62.4**	**53.7**	**57.5**

Table 5: Detection performance when training on GMU Kitchens and testing on AVD. *Synthetic images used in [3] and ours are slightly different.

Method	Accuracy
Random	0.3
BRISK [41]	9.4
ORB [42]	6.6
SURF [43]	10.8
BOLD [44]	7.4
SIFT [26]	12.9
Line-2D [45]	.9
Color Hist [46]	9.2
HMP [47]	25.4
CaffeNet [5]	41.0
CaffeNet+MV[5]	44.1
TDID(ours)	**50.5**

Table 6: One-shot instance classification in a scene.

4.1.2 GMU Kitchens to AVD

We now compare on a different object instance detection task to Faster-RCNN [1]. Dwibedi et al. [3] explore how to create synthetic training data for instance detection, and evaluate how their synthetic data can improve a detector's performance when trained on one dataset, but tested on another. They train/test on the six instances present in both the GMU Kitchens dataset and AVD. In this task, the detector is trained on the GMU data, and tested on all images in the initial release of AVD (17,556 images).

First, we train only on the real images from GMU, and test on AVD. We use the same training hyper-parameters as in the previous instance detection task. On this challenging task TDID is able to outperform Faster-RCNN by over 8 mAP.

Next, we add synthetic images to training. Dwibedi et al. [3] did not release their synthetic images, but did release code to generate them. We use their code and settings described in the paper to generate 5,160 synthetic images ([3] report generating about 6000). Given extra training data, both Faster-RCNN and TDID improve. TDID retains its advantage over Faster-RCNN by 6 mAP, which may be further improved with better synthetic data.

4.2 One-Shot Instance Classification

We have shown our method outperforms state-of-the-art general object detectors on multiple instance detection tasks. We now show that we can also surpass other instance recognition and template matching work, as well as generalize to unseen target instances. Held et al. [5] classify images of instances when given only a single image in training. They show a neural network, combined with some multi-view pre-training, can outperform previous non-deep-learning feature matching methods. Held et al. [5] use a CaffeNet[48] classification network, pre-trained on ImageNet[49]. They then perform a multi-view pre-training step on BigBIRD, train on a single example of each instance in the RGB-D Scenes[8] dataset, and test classification accuracy on crops of instances in RBG-D Scenes.

method	coca cola	coffe mate	honey bunches	hunt's sauce	mahatma rice	nature v1	nature v2	palmolive orange	pop secret	pringles bbq	red bull	mAP
TDID	30.8	73.8	52.0	24.1	26.7	86.1	82.2	28.3	62.2	26.0	37.9	48.2
Faster-RCNN[3]	88.5*	95.5*	94.1*	88.1*	90.3*	97.2*	91.8*	80.1*	94.0*	92.2*	65.4*	88.8*

Table 7: Few-shot detection performance on GMU Kitchens dataset. None of the instances were seen as targets during training, though nature v1 and v2 are very similar to those seen in training. *Faster-RCNN trains on these instances, numbers are just for reference.

We adapt our object detection framework to perform classification, and evaluate this modified network on the same one-shot instance classification task. In this setting, the definition of "target image" stays the same, but "scene image" is now a classification style image, i.e. a crop around one object. For TDID to generalize to unseen target instances it must be provided with a large variety of target instances during training. We construct a training set consisting of over 250 instances from the BigBIRD dataset and RGB-D Object Dataset. More details on training our detection network for classification can be found in the appendix.

For a fair comparison, we use AlexNet [28] (extremely similar to CaffeNet[48], same performance on ImageNet classification) pre-trained on ImageNET as our backbone network. To test how well our system can generalize to unseen target instances, we do not train on the single example of each test instance as [5] does. Instead we use the provided example as the target image at test time, **never re-training our network or updating the weights** to recognize these new objects. Even without any fine-tuning on the test objects our method achieves 50.5% classification accuracy, outperforming the previous deep-learning approach that does train on the test objects, as well as several feature and template matching methods. See Table 6.

4.3 Few Shot Instance Detection

We next explore few-shot instance detection with two examples of each instance available, one front view and one back view. In contrast with usual few-shot tasks, *we do not train on examples of test objects*. We use the examples as target images at test time, requiring our detector to generalize to unseen objects without any on-line training or fine-tuning. High performance on this task could be useful for many applications where the system is given just a few examples of a target object but does not have time to re-train.

We test on the instances on the entire GMU Kitchens dataset. Though some of these are also present in the AVD dataset, we do not ever use them as targets. As in the classification task, we construct a large training set to enable TDID to generalize. We include images from AVD, RGB-D Scenes, synthetic images using RGB-D Objects, BigBIRD, and [3], as well as the ImageNET VID [49] dataset. More details can be found in the appendix.

As shown in Table 7, TDID is able to generalize well to these instances, achieving 48.2 mAP. We also provide the Faster-RCNN results [3] from training/testing on split one of the GMU data as a sort of upper bound reference, and to show the difficulty of the GMU data relative to other tasks. This result is particularly exciting as TDID is able to give reasonable performance on a task general category detectors cannot perform. The ability to detect novel objects quickly, without any new training, could be very valuable for robots in many applications.

5 Conclusion

We propose a new Target Driven Instance Detection method. Using the target driven approach with our feature embedding module, we are able to transform methods for category-level detection to a high performance model for instance detection. We show state-of-the-art performance on multiple instance detection and classification tasks, as well as promising results on a new few-shot detection task. The effectiveness of our embedding allows the model to be lightweight, achieving fast run times without sacrificing accuracy. Future work includes improving generalization to new targets, allowing a model to be used "off the shelf" without retraining for any new target instances.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[3] D. Dwibedi, I. Misra, and M. Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *ICCV*, 2017.

[4] G. Georgakis, A. Mousavian, A. C. Berg, and J. Kosecka. Synthesizing training data for object detection in indoor scenes. In *RSS*, 2017.

[5] D. Held, S. Thrun, and S. Savarese. Robust single-view instance recognition. In *ICRA*, 2016.

[6] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg. A dataset for developing and benchmarking active vision. In *ICRA*, 2017.

[7] G. Georgakis, M. A. Reza, A. Mousavian, P.-H. Le, and J. Kosecka. Multiview rgb-d dataset for object instance detection. In *3DV*, 2016.

[8] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, 2011.

[9] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *ICRA*, 2014.

[10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010.

[11] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. *ICRA*, 2011.

[12] H. Song, S. Zickler, T. Althoff, R. Girshick, M. Fritz, C. Geyer, P. Felzenszwalb, and T. Darrell. Sparselet models for efficient multiclass object detection. *ECCV*, 2012.

[13] A. Collet, M. Martinez, and S. Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *in International Journal of Robotics Research*, 2011.

[14] J. Tang, S. Miller, A. Singh, and P. Abbeel. A textured object recognition pipeline for color and depth image data. *ICRA*, 2012.

[15] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. *ACCV*, 2011.

[16] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. 2018.

[17] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. *CVPR*, 2015.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[20] K. H. J. S. Jifeng Dai, Yi Li. R-FCN: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[22] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017.

[23] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[24] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[25] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016.

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[27] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[29] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017.

[30] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016.

[31] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *PAMI*, 2015.

[32] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[33] K. Chen and W. Tao. Once for all: a two-flow convolutional neural network for visual tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[34] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016.

[35] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016.

[36] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017.

[37] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[38] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[40] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CVPR*, 2017.

[41] S. Leutenegger, S. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *ICCV*, 2011.

[42] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *ICCV*, 2011.

[43] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[44] F. Tombari, A. Franchi, and L. Di. Bold features to detect texture-less objects. In *ICCV*, 2013.

[45] S. Hinterstoisser, C. Cagniart, P. N. N. F. P. Ilic, S. Sturm, and V. Lepetit. Gradient response maps for real-time detection of textureless objects. *PAMI*, 2012.

[46] M. J. Swain and D. H. Ballard. Color indexing. *1991*, IJCV.

[47] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for rgb-d based object recognition. In *ISER*, 2013.

[48] Y. Jia, E. Shelhamer, J. Donahue, Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[49] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[50] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

A Appendix Introduction

In this supplementary material, we give more training details for our experiments in Sections 2-4, as well as a visualization of some networks outputs in Section 5. We have also submitted our code in the supplementary material.

B Training Details for AVD Experiment (4.1.1)

We set the batch size as five, resizing the images from $1920x1080$ to $960x540$, with one sample consisting of two target images and one scene image. We choose training samples such that the target instance is present in the scene image about 60% of the time. This means in about 40% of the training samples, the target object is not present in the scene image and all anchor boxes become negative samples. We start the learning rate as $.001$, momentum as $.9$ and weight decay 0.0005 and train for 40 epochs. We then reduce the learning rate by a factor of 10, and continue training for another 15 epochs.

C Training Details for One-shot Classification Experiment (4.2)

For TDID to generalize to unseen target instances it must be provided with a large variety of target instances during training. We construct a training set consisting of over 250 instances from the BigBIRD dataset and RGB-D Object Dataset. We are careful to not include any instances in training that overlap with those in the test set, RGB-D Scenes, excluding any instances from the test categories: bowl, cap, cereal box, coffee mug, flashlight, soda can. The BigBIRD images are cropped using the provided segmentation masks to produce classification images. The instances from the RGB-D dataset are placed against random background images from the background images of RGB-D Scenes, as in [5].

The instance detection task in previous sections stress finding small objects in large scene images, while in classification the object of interest fills almost the entire image. To enable TDID to recognize these relatively large objects, we adjust the default anchor boxes to be very large, covering almost the entire image. This means instead of predicting classification scores and regression parameters for a large grid of anchor boxes as in Figure 1a, there exist only a few large anchor boxes to be classified.

During training, we ignore any loss associated with bounding box regression, and only focus on classifying the anchor boxes. Since all anchor boxes are large, and the object of interest fills most of the image, we treat all anchor boxes as positive examples when the target image matches the object to be classified. We train with a batch size of 128, and sample training examples such that the target matches the scene image 50% of the time. The learning rate is set at $.001$, momentum $.9$ and weight decay 0.0005 while training for 20,000 iterations. At test time, our classification score for the image is taken as the maximum score of any anchor box.

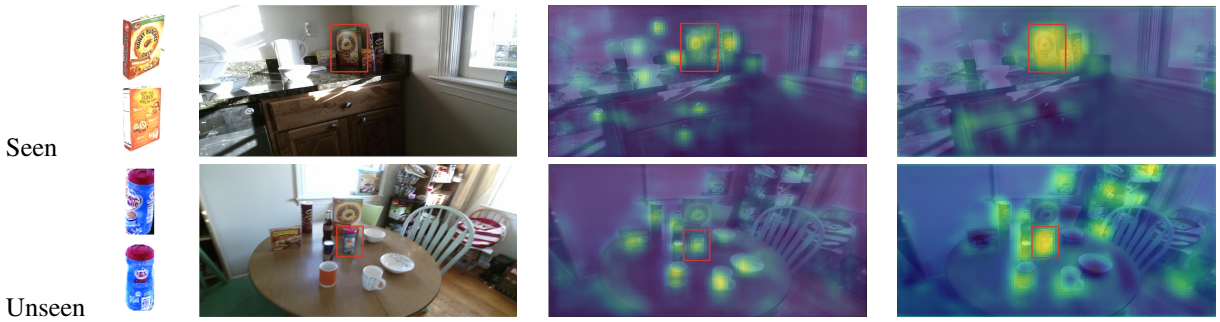


Figure 2: From Left: target images, scene image, VGG activations, our joint representation activations. The top row shows an example where the target was used in the train set, bottom row shows a novel target. The model was successful in detection both objects. Ground truth bounding box in red.

D Training Details for the Few shot detection Experiment (4.3)

In constructing a training set, we use the same instances from BigBIRD and RGB-D Objects as in the classification task, but instead of creating cropped classification images we use the code from [3] to synthetically place the objects in the 1449 images from NYUD2[50].

We further increase the number of target instances seen during training by adding in images from the AVD dataset. In addition to the released bounding box labels for the BigBIRD-like instances, we take advantage of the structure of the AVD dataset to add more target instances to training. We start with an image, I, in a scene, S. Using selective search, we can get the bounding box of some object or region, O, in I. Using the camera locations and depth images provided by AVD, we can project 0 to world coordinates and then project back into every other image in S. This gives us more target instances almost for free. Unfortunately this setup is still experimental, and is not always robust to occlusion and other factors. Therefore we only generate samples from two scenes from AVD, adding a total of about 5000 target/scene image pairs. Future work includes making this process more robust to hopefully greatly increase the generalizability of TDID.

Finally, we add in the ImageNET VID dataset to further increase variety. This dataset consists of snippets of video with one or more objects labeled with a bounding box throughout the video. While training TDID, we first choose a video at random. We then choose an object as the target, and crop two random frames of the video to get target images. Another random frame of the video is chosen as the scene image 50% of the time, while a random frame from a random video is chosen the other 50%. This means the target object is visible in the scene image in half of the training examples.

We use the same training hyper-parameters as in the detection experiment on AVD in Section 4.1, except we cut the learning rate in half to $.0005$ and train for 150,000 iterations.

E Visualization

Figure 2 shows a visualization of activation responses from hidden layer neurons in our network. To visualize these activations we take the average value across all channels in each spatial location. This corresponds to finding locations where a large number of neurons are activated. We analyze two heatmaps: VGG features (after fine-tuning) from the scene image and the joint representation that is directly fed into the detection prediction head.

Two scene-target pair cases are demonstrated. In the first case the target was seen during training. VGG features show that neurons fire in locations where objects are present. Notice how the activations change when going to the joint representation: a clear peak corresponds to the actual target location.

The second target was not seen during training. As in the previous case, VGG activations highlight various objects present in the scene and a medium activation is visible on the actual target. It is not easy to distinguish between the target and other objects. The joint representation magnifies

activations on the objects that are similar to the target and the peak activation blob highlights the actual target, although the difference in activations is expectedly weaker.