




YADA: you always dream again for better object detection

Khanh-Duy Nguyen¹  · Khang Nguyen¹ · Duy-Dinh Le¹ · Duc Anh Duong¹ · Tam V. Nguyen²

Received: 12 September 2018 / Revised: 26 April 2019 / Accepted: 10 June 2019 /

Published online: 8 July 2019

© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Object detection has been attracting a lot of attention from the computer vision community. It has a wide range of practical applications ranging from the traditional use such as image annotation to modern uses such as self-driving vehicles, robotics, surveillance systems, and augmented reality. Recently, deep learning has significantly improved the state-of-the-art performance of the object detection task. Many works explore various deep network structures to improve the performance. However, the impact of training data is still not well investigated. Although some works focus on data augmentation and data synthesis, there is no guarantee that they are effective for the training process. In this paper, we propose a novel framework addressing the problem of generating relevant data and how to use them effectively. We apply lucid data synthesizing which generates data by mining hard examples and embedding them to the same context locations. Further, we utilize a dual-level deep network leveraged with these generated data to effectively detect hard objects in images. Extensive experiments on two benchmarks, PASCAL VOC and KITTI, demonstrate the superiority of our approach over the state-of-the-art methods.

Keywords Object detection · Deep learning · Data synthesis

✉ Khanh-Duy Nguyen
khanhnd@uit.edu.vn

Khang Nguyen
khangntm@uit.edu.vn

Duy-Dinh Le
duyld@uit.edu.vn

Duc Anh Duong
ducda@uit.edu.vn

Tam V. Nguyen
tamnguyen@udayton.edu

¹ University of Information Technology, VNU-HCM, Viet Nam

² University of Dayton, Dayton, OH, USA

1 Introduction

Object detection is one of the most important tasks in computer vision through this decade. It aims to predict the existence of objects and localize the objects in a given image. This is very useful in a wide range of applications, e.g. self-driving vehicles, robotics, and augmented reality. In the earliest stage of object detection, pioneering works utilize hand-crafted features for object representation. Viola et al. utilize the lightweight Haar features and cascade classifier to efficiently detect human faces [48]. Later, Dalal et al. introduced Histogram of Gradients (HOG) [5] as effective features for pedestrian detector. HOG is then widely used to detect other objects. Felzenszwalb et al. proposed discriminatively trained part based models (DPM) [9] to detect the objects with deformable parts, i.e., humans with different poses. Van de Sande et al. proposed Selective Search [46] method which uses segmentation to generate a limited set of locations, permitting the more powerful yet expensive bag-of-words features. Wang et al. proposed a hierarchy representation of low-level features in local regions, named Regionlets [49], which is more robust to generic deformable objects. Recently, the advancement in deep learning and Convolution Neural Network models (CNN), i.e., Region CNN [13], Fast RCNN [12], Faster-RCNN [33], R-FCN [4], SSD [27], and YOLO [32], significantly boosts the performance of the object detection task. Modern object detection approaches shift from designing features and object representation models to applying different architectures of neural networks. There exist many works exploring different deep network structures to improve the performance, such as AlexNet [24], VGGNet [37], GoogLeNet/Inception [39–41], ResNet [17, 26], DenseNet [19]. Convolutional neural networks have become deeper and deeper, with state-of-the-art networks going from 7 layers (AlexNet) to 1000 layers (ResNet). Obviously, deep networks require much more computing resources, i.e. GPU memory, and run slower than shallow ones in general. Our main objective in this paper is not only limited in the deep structure context. Instead, we aim to investigate the impact of training data in the deep networks. In literature, several works have focused on enriching data, i.e. data augmentation [27, 32, 33] or generating synthetic datasets [10, 16, 35, 47]. However, these works consider the amount of generated data rather than the importance of these data, that is which objects should be generated more and how to use them effectively.

In psychology research, the term “lucid dreaming” is used to describe the technique of controlling dreams and following them to a desired conclusion [22]. Additionally, lucid data are the images that the dreamer is aware that they are dreaming. In this work, we consider lucid data as the synthesized data for the specific problem, object detection. Our “lucid dreaming” is the process of synthesizing desired training data to train an object detector. It generates new training data by discovering the previously failure cases of the object detector. Then, these failure cases are synthesized onto many related scenes to strengthen the detector. In particular, we explore intentional synthesized data used for training a deep learning model and propose a following effective detection strategy. Our proposed framework is named YADA (the short form of **Y**ou **A**lways **D**ream **A**gain), which represents the idea of using lucid data dreaming, then re-train a deep model for a better detection performance. We introduce our novelty in two stages, namely, data preparation during pre-training, and data residual on the post-training. Regarding the data preparation, we propose using lucid data dreaming in order to produce more problem-related training data. For the data residual, we first train a detection model, here we adopt Faster-RCNN [33] for our work. Then we train another Faster-RCNN model to tackle the challenging objects.

The remainder of our paper is organized as follows. Section 2 summarizes the related works. Section 3 introduces the proposed framework. Section 4 presents the experimental results. Finally, the conclusion and future works are given in Section 5.

2 Related works

Recently, the advancement in deep learning significantly improves the performance in many computer vision problems. For instance, CNN [25] improves the performance of image recognition, image parsing, and saliency analysis [42–44]. The success of CNN model for these tasks has inspired other works for integrating CNN model into the object detection problem, i.e., Region CNN [13], Fast RCNN [12], Faster-RCNN [33], R-FCN [4], SSD [27], and YOLO [32]. Several works have succeeded in further boosting the performance of CNN object detectors by applying advanced detection techniques. Cheng et al. [1, 2] proposed an effective method to train rotation-invariant and Fisher discriminative CNN (RIFD-CNN) models. This method can improve the performance of R-CNN [13], Fast R-CNN [12], Faster R-CNN [33], and R-FCN [4]. Chu et al. [3] proposed multi-scale adjacent level feature maps and cascaded region proposal network for improving both recall and accuracy of Fast R-CNN and Faster R-CNN detectors. Zhang et al. [51] presented a novel weakly supervised learning framework which utilizes instance-level and image-level prior-knowledge for object detection. Next, SPFTN [50] is proposed for addressing the weakly supervised video object localization and segmentation tasks. Several approaches focus on designing features and classifiers to applying different architectures of neural networks. Accordingly, lots of networks, such as AlexNet [24], VGGNet [37], GoogLeNet/Inception [39–41], ResNet [17, 26], DenseNet [19], have been explored. A comprehensive review for advanced deep network object detectors is presented by Han et al. [15]. Convolutional neural networks have become deeper and deeper, with state-of-the-art networks going from 7 layers (AlexNet) to 1000 layers (ResNet). Generally, deep networks give better accuracy than shallow ones thanks to their advantage on the approximation of compositional functions [34, 52]. The limitation of deep networks is requiring much more time for training and testing. Several works success in designing of a light-weight network that can achieve the competitive accuracy, for instance, SqueezeNet [20], Darknet-19 [30] and Darknet-53 [31].

The aspect of data on training deep network structure is also explored. In a recent paper, Ross Girshick et al. [28] scaled up to 3.5 billion images and 17,000 distinct “labels” for the training process and get the successful result. Data is costly and therefore people have to find the way to augment the available labelled data. Basic data augmentation techniques such as cropping, flipping, and colour jittering are commonly used in many works (Faster-RCNN [33], YOLO [32], SSD [27]). This helps generate more samples for object class by applying transformations and then helps the trained detectors can recognize objects through minor changes in appearance. Other efforts come from the idea of augmenting data by GANs model.

From another viewpoint, it is not only the amount of supplied data that is most important. The quality (or reasonability) of data is also the matter. To deal with this, several works focus on generating realistic synthetic data (which try to make synthetic objects having the same appearances, locations, without major artefacts). Rendering images from 3D models is a commonly used method for generating synthetic datasets, e.g. SYNTHIA [35], SceneNet [16], Virtual KITTI [10], SURREAL [47]. This approach is also used for rendering realistic data for training object detector. Gupta et al. [14] use 3D CAD objects models

and render them into scenes. They observe an 1% improvement mean AP point on NYUD2 dataset comparing with without using synthetic data. As a closer look, Peng et al. [29] investigate the ability of CNN object detector to learn from synthetic CAD-rendered images with/without simulating low-level cues, such as realistic object texture, pose, or background. Likewise, Tremblay et al. [45] rendering 3D car models with varying aspects of the scene (e.g. car type, texture, location, camera angle, and lighting). Their experiments on KITTI dataset show the performance of Faster R-CNN detector with additional training on the synthetic data superiority the one using COCO initialized weights. Johnson-Roberson et al. [21] leverage the rich virtual worlds created for major video games to create synthetic data. They captured images from the video game with different simulated times of day, complex weather, and lighting scenarios.

The rendering-based data are expensive to generate, requiring artists to carefully model specific environments in detail, and typically limited to a few categories like cars. Without the advent of rich 3D repositories, Dwibedi et al. [6] proposed a cut and paste method to generate synthetic images. Then, they trained a Faster R-CNN detector using the VGG network. Their method, when combined with real images, improves relative performance by more than 21% on GMU Kitchen dataset. The success of this approach should encourage more inventing on popular object detection datasets such as PASCAL VOC and KITTI. However, there is no successful result reported until now. Our hypothesis is this due to the unintentional mode of generating synthesized data. Many easy objects could be learned effectively by deep networks, and therefore generating more instance of these objects cannot improve the performance of the system. The synthesizing process should focus on the hard and usually rarely appearing objects. Then we proposed a method to create intentional synthesized data by adopting hard example mining, as inspired by our previous work [23]. Instead of generating more object instances, we try to explore the hard and rarely appearing cases in the dataset and then generate synthesized data for these cases. Furthermore, we build a separate synthesized set for these cases and then train an additional specific detector. Detection results from this detector can effectively complement for the detector trained by real data.

3 Proposed framework

In this section, we will introduce the proposed YADA framework in details. Figure 1 shows the overview of our framework pipeline.

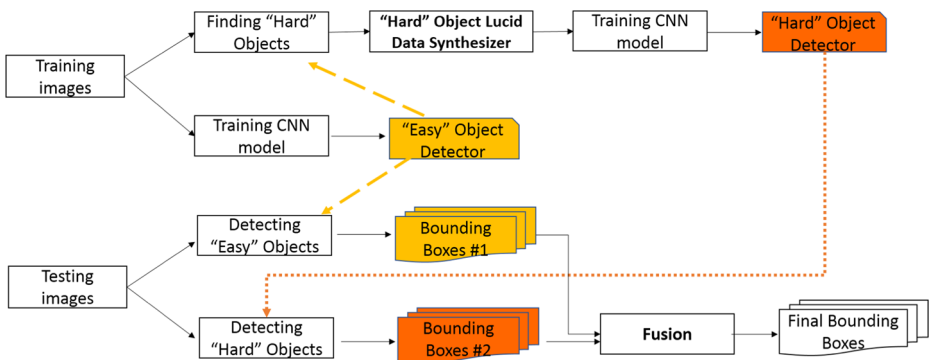


Fig. 1 Our YADA framework pipeline

3.1 Lucid data synthesizer

3.1.1 Similar scene retrieval

In this context, we define the “easy” and “hard” objects as the objects detected and misdetected by a trained object detector, respectively. “Hard” objects are discovered depending on the baseline detector. In other words, we train the baseline detector, then use the trained model to detect objects in training image set. Objects misdetected by the baseline detector (or have confidence score lower than the detection threshold) are considered as “hard objects”. The first step to synthesize hard object instances data is to create a pool of similar images. This step aims to help the synthesized data have the real contexts as the original one. Given the training set images T , for each image I_q in T we use the features extracted from the layer fc_7 of the VGG imagenet pre-trained model to retrieve the top k similar images in the training set to form its similarities pool. Let P denote the pool of similar images (a ranked list of images). Let V_q and V_i^P denote the feature vectors of the query image I_q and of the image I_i from the pool, respectively. Then we define the similarity level between I_q and the i -th image of P as the Euclidean distance between their corresponding features vectors: $s_i = \|V_q - V_i^P\|$. The smaller the Euclidean distance is, the higher level the similarity of the two images is. Each image I_i is ranked in the ascending order by the similarity.

Note that if the value of k is small, we only paste hard objects to a few images that are very similar to the original image. The method prefers to generate realistic synthesized image rather than increase the number of samples by a significant amount. On the other hand, if we set k by a large value, the similarity of original and destination images will be decreased but we can generate more hard samples for the synthesized set. The value of this parameter is empirically set at 100 in our experiments.

3.1.2 “Hard” object lucid data synthesizer

For lucid data synthesizer, we replace “easy” objects by the “hard” ones to ensure the similar context for the new synthesized image. Specifically, for an image I in the training set, firstly we find all available positions, that are the bounding boxes of detected objects. Next, we create a pool of hard objects by collecting them from top k similar images of I . In particular, we run a Faster RCNN detector on these images to find misdetected objects. The similarity between two images is defined as the Euclidean distance between their corresponding features vectors, as presented in Section 3.1.1. Then, each available position in the image I is matched with an object in the pool of “hard” objects. We exploit the *width*, the *height*, and the *aspect* ratio (*height/width*) measurements of the available boxes and the object bounding boxes for matching. The details are shown in Algorithm 1. The illustration of this process is shown in Fig. 2

Following the synthesizing, we further highlight image regions on the synthesized image set. In particular, we apply histogram equalization for images to make the hard object instances more outstanding. The details of this operation are shown as below equations,

$$\begin{cases} C_{eq(x,y)} = (L - 1) * \tau(C_{(x,y)}) \\ C_{eq} = \{C_{eq(x,y)} : 0 \leq x < W, 0 \leq y < H\}, \end{cases} \quad (1)$$

where $\tau(C_{(x,y)}) = \sum_{i=0}^W \sum_{j=0}^H (C_{(i,j)} \leq C_{(x,y)}) / (W \times H)$, C is a color channel (R/G/B), C_{eq} represents the color channel after histogram equalization, $C_{(x,y)}$ represents the value of

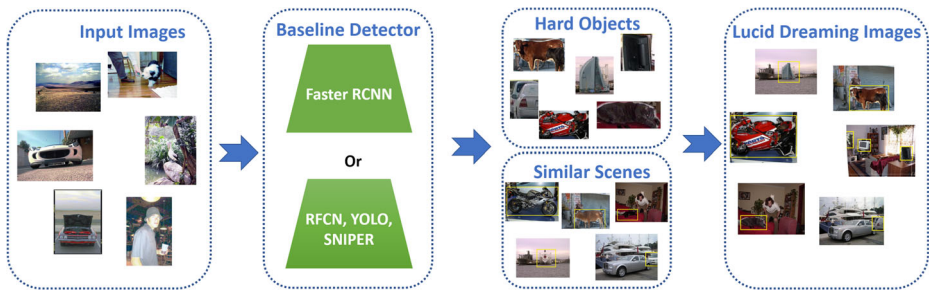


Fig. 2 Lucid Data Synthesizer: the process of generating synthesized images in our framework

pixel (x, y) in the image channel, $C_{eq(x,y)}$ represents the value of pixel (x, y) in the image channel after histogram equalization, L is fixed at 256, W and H are the width and the height of the image, respectively.

Algorithm 1 “Hard” object lucid data synthesizer.

```

procedure SYNTHESIZER( $T$ )
 $T_{syn} \leftarrow \emptyset$ 
for an image  $I$  in  $T$  do
   $L \leftarrow$  All available locations in  $I$ 
   $S \leftarrow$  Set of  $k$  similar images
   $O \leftarrow$  Hard objects in  $S$ 
  for a location  $L_i$  in  $L$  do
     $D \leftarrow$  Similarity distances between  $L_i$  and objects in  $O$ 
     $O_{closest} \leftarrow$  Object in  $O$  which has smallest distance
    if  $D_{O_{closest}} > threshold$  then
       $I_{syn} \leftarrow$  Paste  $O_{closest}$  into  $I$  at  $L_i$  location
    end if
  end for
   $T_{syn} \leftarrow T_{syn} \cup I_{syn}$ 
end for
end procedure

```

Directly pasting objects onto background images obviously creates boundary artifacts. Although these artifacts seem subtle, when such images are used to train detection algorithms, they give poor performance as proved in the work [6]. As current detection methods [33] strongly depend on local region-based features, boundary artifacts substantially degrade their performance. The blending step will smoothen out the boundary artifacts between the pasted object and the background, therefore, they improve performance of the trained detectors. We use the traditional Gaussian blending for smoothing edges. Furthermore, Mask RCNN [18] is also used for segmenting of objects from the background before pasting them on the similar images. Excluding background pixels in object bounding boxes will make synthesized objects better blended with the new scenes.

3.2 Bounding box fusion

For the testing procedure, an input image is fed into the first CNN model to detect “easy” objects. Then, the second CNN model trained for the hard objects is used to detect the undiscovered objects. In fact, the detected objects of the first CNN model could be erased from the images by using the masking to prevent the duplicate detections. However, this manipulation could drop down the recall rate because it also removes a number of objects that are occluded by the detections. Therefore, we propose a fusion method to combine the two bounding boxes set B_1 (first CNN model) and B_2 (second CNN model) as follows. Figure 3 shows the pipeline of proposed fusion scheme.

3.2.1 Duplicate truncation

Firstly, all the duplicated bounding boxes are truncated. These duplicates are identified as follows.

$$D = \{(b_i, b_j) : \frac{area(b_j \cap b_i)}{area(b_j \cup b_i)} > \zeta\}, \tag{2}$$

where b_i is a bounding box in B_1 , b_j is a bounding box in B_2 , and ζ is the NMS threshold and fixed at 0.7. Meanwhile, \cap denotes the intersection operation of two bounding boxes, and \cup denotes the union operation of two bounding boxes.

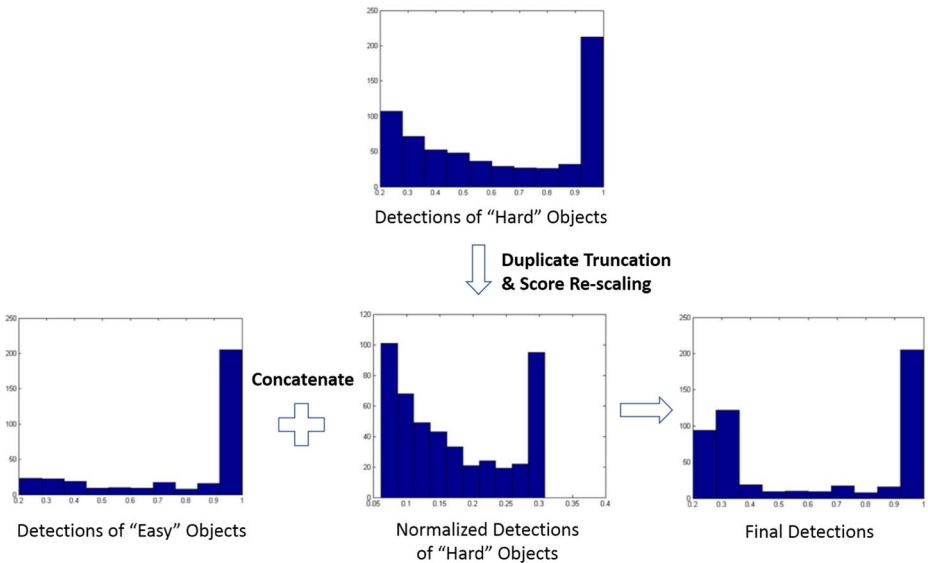


Fig. 3 Proposed detection fusion scheme illustrated by real data (detections of ‘plane’ class - VOC 2007 testing set). The CNN model trained by synthesized data (focusing on hard objects) has a balanced detection confidence histogram whereas most of detections of the CNN model trained by standard data have confidence scores in range [0.9-1.0]. Our fusion method improves the detections recall by normalizing the detection of the synthesized data based model (including duplicates truncating and scores rescaling), then concatenating both of detections set

We remove the duplicated bounding boxes from the B_2 instead of B_1 because the detections from the second model generally have lower confidence scores.

3.2.2 Score re-scaling

We then rescale confidence scores of detections from the second model. This aims to two purposes: firstly, we want to handle the uncertainty of these detections. These detections focus on hard objects therefore their confidence should not be higher than the confidence of easy objects. Second, we need to scale these scores to an reasonable range so that they can effectively complement the detections from the first model.

To do that, for each object class C we estimate the mean value μ_C of detection scores from the first model. Then we rescale confidence scores of detections from the second model by the following equation:

$$S^{Normalized} = S^{Original} \times \frac{1}{\mu'_C} \times (1 - \mu_C - \gamma \times \sigma_C), \quad (3)$$

where μ'_C is the mean value of detection scores from the second model, σ_C is the standard deviation of detection scores from the first model, and γ is the coefficient for flexible handling the fusion of two Gaussian distributions. The idea behind this Eq. 3 is that we want to re-scale the confidence scores of the hard object detector by translating their mean values. The new mean value is determined based on the mean value of confidence scores from the baseline detector. γ is a coefficient score to establish the distance between two mean values. For experimental settings, we set $\gamma = 0.2$.

4 Experimental results

4.1 Benchmark datasets

We evaluate our proposed method on two challenging real-world datasets: PASCAL VOC and KITTI. The PASCAL Visual Object Classes Challenge [7] is a popular dataset in object detection and recognition. The KITTI dataset [11] is used as a benchmark for autonomous driving systems.

We conduct experiments on the PASCAL VOC 2007 dataset with 9,936 images and 24,640 annotated objects of 20 classes including people, vehicles, animals, and indoor objects. Half of them is used for training and validation process, and the other half is used for testing. The number of objects in each class is approximately equivalent in both sets. We also make the statistic comparison of hard objects/total objects ratio between the original trainval set and our synthesized trainval set in Fig. 4. Visualization of synthesized images is shown in Fig. 5.

The KITTI dataset has 7,481 images in the training set and 7,518 images in the test set with a total of 80,256 labelled objects of 8 classes: pedestrian, car, van, truck, tram, and misc. Yet, ground-truth labels are available only on the training set.

For measuring the performance, we use the average precision (AP) metric for both of the two datasets. This is the average of the precision obtained at all values of recall corresponding with the ranked output bounding boxes. We take the method of computing AP using all

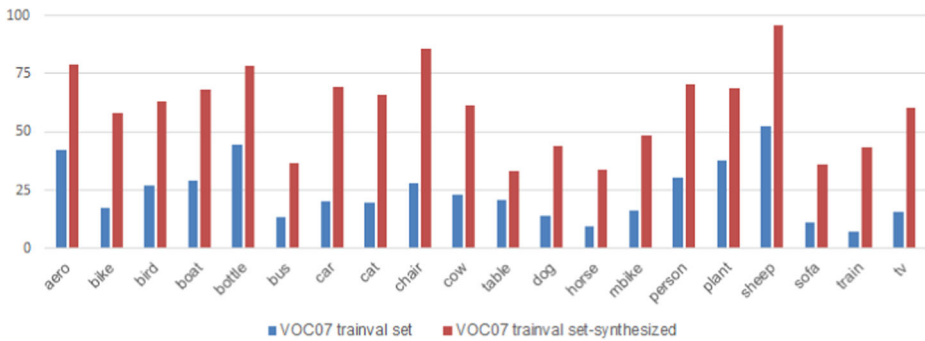


Fig. 4 Statistic comparison of hard objects/total objects ratio between VOC 2007 train-val set and synthesized VOC 2007 train-val set

data points [8] instead of using only 11 points with equally spaced recall as in [7]. For more details, the precision and recall are defined as follows:

$$precision = \frac{tp}{tp + fp} \quad (4)$$

$$recall = \frac{tp}{tp + fn}, \quad (5)$$

where tp denotes the number of detected bounding boxes which are correct, fp denotes the number of detected bounding boxes which are incorrect, and fn denotes the number of missed bounding boxes.

A bounding box is decided as whether true positive or false positive by measuring the ratio between its intersection area and union area (with a ground-truth box), which usually called IoU (Intersection over Union). This metric is computed as the equation below:

$$\tau = \frac{area(b_p \cap b_g)}{area(b_p \cup b_g)}, \quad (6)$$

where b_p is a predicted bounding box and b_g is a ground-truth bounding box, \cap denotes the intersection operation of bounding boxes, and \cup denotes the union operation of bounding boxes. In order to be considered as a true positive, τ must be greater than 0.5. A predicted box will be assigned to the best-overlapped ground-truth box (which has the largest IoU). When multiple boxes overlap with a ground-truth box, that means all of these boxes satisfy the constraint of IoU, only the box which has the maximum IoU is accepted as true positive, the leftovers are judged as false positives.

4.2 Implementation settings

We conduct experiments on the medium scale CNN model - VGGM and the large scale one - VGG16. We also report the performance of our reproduced Faster-RCNN in order to make a fair comparison. Note that for implementing Faster-RCNN, we modify the author's public source code¹ to work well with the OHEM-model,² which is developed for Fast RCNN. This model can significantly reduce the consumed GPU memory by using gradient accumulation over two forward/backward passes. Thus, we can run the VGG16 network on

¹<https://github.com/rbgirshick/py-faster-rcnn>

²<https://github.com/abhi2610/ohem>

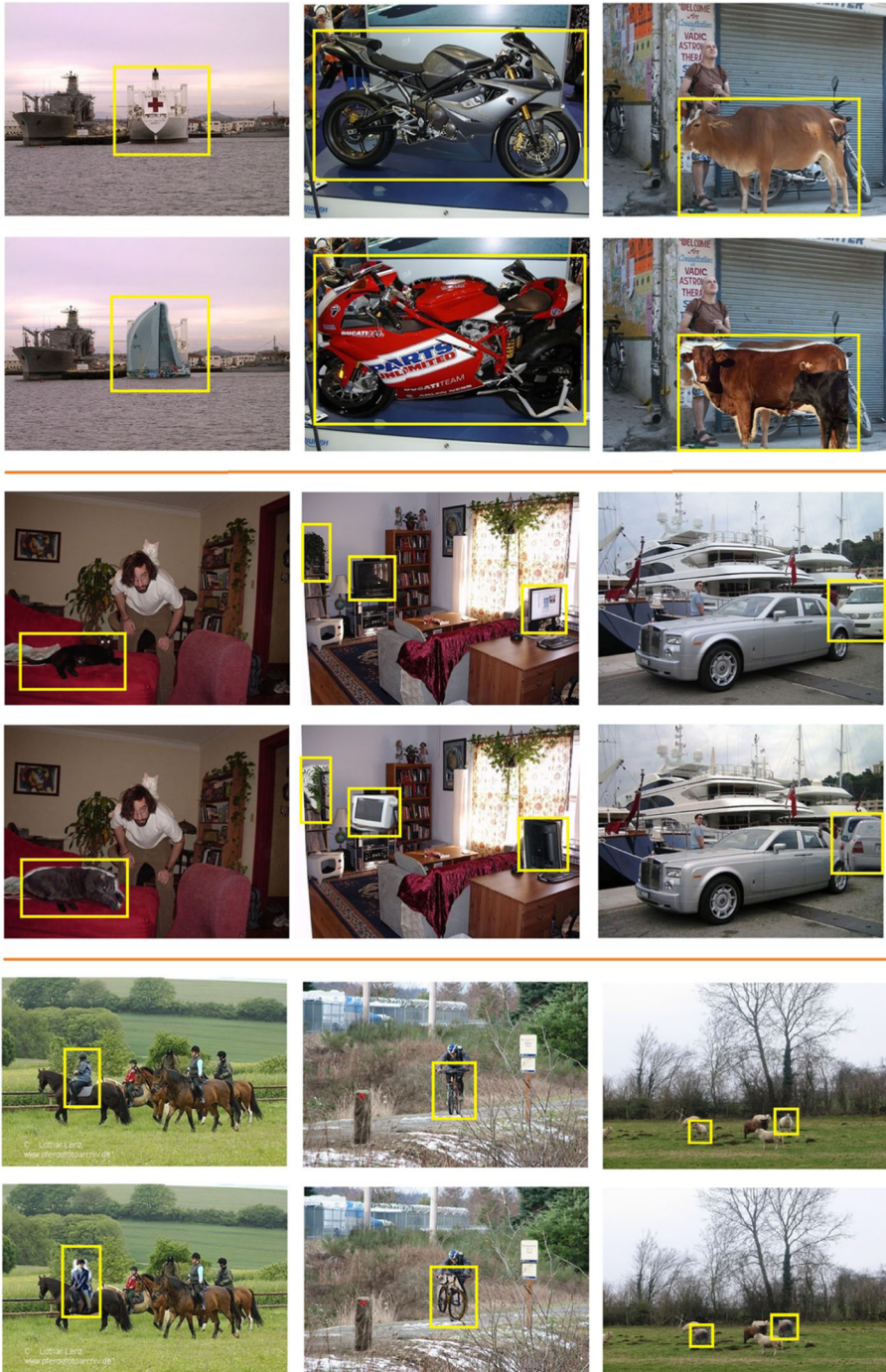


Fig. 5 Synthesized images from VOC 2007 dataset. The first row of each group (separated by the red line) shows real images in the dataset, whereas the second row shows the corresponding synthesized images. Replaced objects are highlighted by yellow rectangles

Table 1 Detection results on PASCAL VOC 2007 test set (all methods are trained on VOC 2007 trainval set)

Model	Faster-RCNN [33]	Faster-RCNN*	YADA
VGGM - SS	61.8	61.7	63.2
VGGM - MS	N.A.	63.0	64.0
VGG16 - SS	69.9	70.4	72.5
VGG16 - MS	N.A.	71.9	73.5

Evaluation metric is AP (AUC). MS: multi-scale, SS: single scale. N.A. is for the null results where the results cannot be found in the original paper. * indicates our own Faster-RCNN implementation. The best performance of each model is marked as boldfaced

the medium GPU card (Tesla K20c), which requires maximum 4GB of RAM. However, the disadvantage is that this supports only alternating optimization training and therefore taking more training time.

4.3 Performance on PASCAL VOC

The experimental results of different network settings are shown in Table 1. Regarding the single scale setting, our YADA improves the Faster-RCNN by a large margin. With VGG16 network backbone, YADA outperforms Faster-RCNN by **2.6%**. In addition, multi-scale is also proved its superiority over single-scale Faster-RCNN setting with 1.0% gain.

We also compare our proposed method with other state-of-the-art methods. The results in Table 2 show the effectiveness of YADA. Our method outperforms all the baselines. In details, YADA outperforms the Fast RCNN by **5.4%** mAP, and the Faster-RCNN by **3.6%** mAP. Comparing with the online hard example mining - OHEM method [36], our method achieves an improvement by **3.6%** mAP. We also evaluate our proposed method

Table 2 VOC 2007 test detection average precision (%)

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow
Fast-RCNN	74.6	79.0	68.6	57.0	39.3	79.5	78.6	81.9	48.0	74.0
Faster-RCNN	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3
OHEM	71.2	78.3	69.2	57.9	46.5	81.8	79.1	83.2	47.9	76.2
SSD300	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8
SSD512	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0
YADA-SS	73.0	83.3	72.4	58.1	51.5	83.1	85.5	87.5	50.2	78.5
YADA-MS	75.5	84.7	74.3	61.2	54.0	83.2	87.1	85.6	53.1	80.3

Method	mAP	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast-RCNN [12]	68.1	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5
Faster-RCNN [33]	69.9	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
OHEM [36]	69.9	68.9	83.2	80.8	75.8	72.7	39.9	67.5	66.2	75.6	75.9
SSD300 [27]	68.0	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5
SSD512 [27]	71.6	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8
YADA-SS	72.5	68.7	84.1	86.1	77.9	80.3	38.7	72.7	65.7	77.9	74.3
YADA-MS	73.5	68.7	83.3	85.1	80.4	80.6	42.2	72.0	65.2	77.9	76.4

All methods use VGG16 network. Training set is VOC07 trainval. MS: multi-scale, SS: single scale. The best performance of each category is marked as boldfaced

Table 3 VOC 2007 test detection average precision (%)

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow
Fast-RCNN	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8
Faster-RCNN	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9
OHEM	77.7	81.2	74.1	64.2	50.2	86.2	83.8	88.1	55.2	80.9
SSD300	75.5	80.2	72.3	66.3	47.6	83.0	84.2	86.1	54.7	78.3
SSD512	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1
YADA-SS	81.6	85.3	78.7	65.8	59.0	88.5	89.2	93.3	56.6	85.5
YADA-MS	82.0	85.4	82.0	70.1	63.0	89.2	90.1	92.4	58.6	88.3

Method	mAP	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast-RCNN [12]	70.0	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster-RCNN [33]	73.2	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
OHEM [36]	74.6	73.8	85.1	82.6	77.8	74.9	43.7	76.1	74.2	82.3	79.6
SSD300 [27]	74.3	73.9	84.5	85.3	82.6	76.2	48.6	73.9	76.0	83.4	74.0
SSD512 [27]	76.8	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
YADA-SS	77.9	71.0	88.9	89.9	80.6	83.1	44.1	80.9	73.4	84.2	79.0
YADA-MS	79.3	72.0	88.4	89.5	80.5	84.4	46.8	82.1	74.8	85.9	80.2

All methods use VGG16 network. Training set is VOC07+12 trainval. MS: multi-scale, SS: single scale. The best performance of each category is marked as boldfaced

with the training data from PASCAL VOC 2007 and 2012. As shown in Table 3, all methods improve their performances owing to the larger training data. Again, the YADA method leads all baselines with a remarkable margin. Figure 6 shows some visualization results of our YADA method on the PASCAL-VOC 2007 test set. Indeed YADA well discovers the unseen (challenging or undetected) objects.

4.4 Performance on KITTI

For KITTI dataset, we evaluate our work on the validation set due to the unavailability of the ground-truth on the test set. We split $\frac{2}{3}$ released images for training set and $\frac{1}{3}$ images for validation set. The results of different settings on KITTI dataset are reported in Table 4. Our proposed framework surpasses the Faster-RCNN baselines in all network structures. As a closer look in VGG16 implementation, our proposed method significantly improves Faster-RCNN in the three main categories, namely, car, pedestrian and cyclist by **2.8%**, **3.7%**, and **2.5%**, respectively. This improvement highlights the potential usage of our method into the practical systems such as autonomous vehicles. The overall performance of YADA-SS is **83.2%**, that improves Faster-RCNN-SS with **3.4%** gain. YADA-MS further boosts YADA-SS with 1.8% increment.

4.5 Effect of score-scaling process in YADA

In this subsection, we conduct a comparison experiment on the score re-scaling process, which is used to combine detected objects from the the two models. To make clear the effect of this process, we evaluate performance of our YADA method in two settings: with and without using score-scaling. The results are shown in Table 5. We observe that the

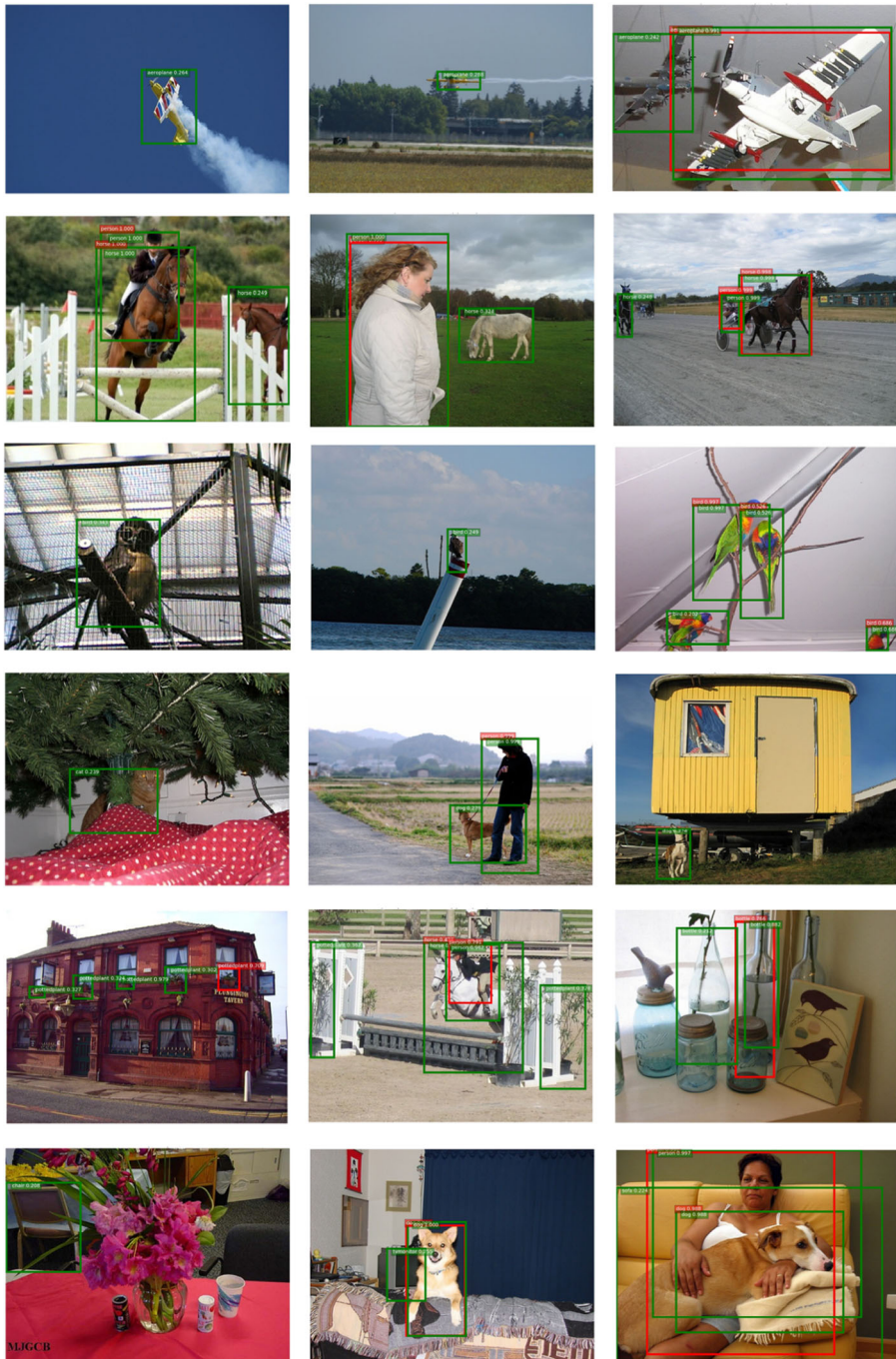


Fig. 6 Visualization of true detections of YADA on VOC 2007. For each image, the red color boxes represent detections from the usual Faster-RCNN model and the green color boxes represent detections from our proposed YADA framework. The result shows the superiority of YADA over the Faster-RCNN with more objects can be detected in images

Table 4 KITTI detection average precision (%) on the validation set for 7 object categories with different network structures of Faster-RCNN and our method

Method	Model	mAP	Car	Pedes.	Cyclist	Truck	Van	Tram	Misc
Faster-RCNN*-SS	VGGM	74.1	80.4	56.9	66.4	86.2	81.4	81.6	65.9
Faster-RCNN*-MS	VGGM	75.9	83.4	58.9	65.9	89.3	83.1	81.1	69.8
Faster-RCNN*-SS	VGG16	79.8	82.5	62.6	72.1	92.2	87.2	88.4	73.5
Faster-RCNN*-MS	VGG16	83.0	87.0	67.1	75.0	94.1	89.2	92.1	76.3
YADA-SS	VGG16	83.2	87.1	68.2	75.9	93.4	89.8	90.7	77.2
YADA-MS	VGG16	85.0	89.8	70.8	77.5	94.7	90.6	92.6	79.1

* indicates our own Faster-RCNN unshared implementation. The best performance of each category (Faster-RCNN or our method) is marked as boldfaced

mAPs of YADA without score-scaling decreased by 2.8% and 3.2% respectively to single-scale and multi-scale configurations. The detections from hard object detector (the second model) should be post-processed before concatenating with the detections from the baseline detector (the first model). Our solution is to re-scale their confidence scores to eliminate noise from uncertain detections. Our YADA method with score-scaling process improves

Table 5 VOC 2007 test detection average precision (%) of YADA with/without score-scaling process

Method		aero	bike	bird	boat	bottle	bus	car
Faster-RCNN*-SS		79.9	83.4	77.2	64.9	57.4	88.0	88.1
Faster-RCNN*-MS		80.4	84.3	80.9	69.5	62.3	89.0	89.2
YADA-SS	without score-scaling	76.8	81.3	72.1	59.9	56.1	85.9	86.7
YADA-MS		79.2	81.5	74.2	61.8	59.1	87.2	88.1
YADA-SS	with score-scaling	81.6	85.3	78.7	65.8	59.0	88.5	89.2
YADA-MS		82.0	85.4	82.0	70.1	63.0	89.2	90.1
Method		cat	chair	cow	table	dog	horse	mbike
Faster-RCNN*-SS		92.9	55.8	83.6	70.8	87.5	89.1	78.2
Faster-RCNN*-MS		91.8	57.9	87.3	71.9	87.8	89.1	79.2
YADA-SS	without score-scaling	89.0	51.8	81.2	68.5	84.0	87.3	73.7
YADA-MS		89.2	53.0	84.5	69.1	84.1	86.8	73.8
YADA-SS	with score-scaling	93.3	56.6	85.5	71.0	88.9	89.9	80.6
YADA-MS		92.4	58.6	88.3	72.0	88.4	89.5	80.5
Method		mAP	person	plant	sheep	sofa	train	tv
Faster-RCNN*-SS		76.7	81.2	42.4	79.3	72.1	83.2	78.4
Faster-RCNN*-MS		78.5	83.1	45.8	81.5	73.8	85.1	79.3
YADA-SS	without score-scaling	73.9	79.1	42.2	75.7	69.8	80.9	75.8
YADA-MS		75.3	80.6	43.9	77.9	70.8	82.7	77.5
YADA-SS	with score-scaling	77.9	83.1	44.1	80.9	73.4	84.2	79.0
YADA-MS		79.3	84.4	46.8	82.1	74.8	85.9	80.2

All methods use VGG16 network. Training set is VOC07+12 trainval. The best performance of each category is marked as boldfaced

Table 6 VOC 2007 test detection average precision (%) of YADA on other baselines

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow
YOLOv2 352x352	72.1	80.3	66.6	56.0	41.1	80.5	79.6	85.5	46.3	69.1
RFCN	82.8	88.4	83.2	71.9	70.3	88.6	90.7	91.7	67.8	88.6
SNIPER	91.3	93.8	88.3	81.2	78.3	91.3	95.1	89.1	75.7	89.6
YADA-YOLOv2	72.5	81.0	67.7	56.6	42.0	80.6	80.4	86.0	47.6	69.6
YADA-RFCN	84.5	89.3	83.5	72.3	71.2	89.2	91.3	91.8	68.1	89.3
YADA-SNIPER	91.4	93.9	88.5	81.4	78.9	91.4	95.2	89.1	76.3	89.5

Method	mAP	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
YOLOv2 352x352 [30]	70.2	72.2	80.5	84.7	83.5	71.8	39.8	66.4	73.9	81.3	72.4
RFCN [4]	82.0	75.6	92.4	89.8	85.4	85.2	52.5	82.9	81.8	88.6	80.9
SNIPER [38]	86.7	84.8	86.8	91.6	93.1	91.5	67.4	86.9	86.7	90.1	80.9
YADA-YOLOv2	70.9	73.1	80.7	85.0	83.9	73.0	41.0	67.3	75.1	82.1	73.1
YADA-RFCN	82.6	75.7	92.7	90.0	86.4	86.3	53.3	83.0	82.1	89.2	81.8
YADA-SNIPER	87.0	85.1	87.2	92.0	93.2	91.7	68.1	87.0	86.9	90.7	81.5

RFCN and SNIPER use Resnet-101 network. YOLO uses DarkNet-19 network. Training set is VOC07+12 trainval. The best performance of each category is marked as boldfaced

Faster-RCNN by 1.2% and 0.8% in terms of mAP corresponding to single-scale and multi-scale configurations.

4.6 YADA on other deep networks

In this subsection, we focus on investigating the impact of our synthesized data on other deep networks. In addition to Faster-RCNN as in the previous experiments, YOLOv2 [30], RFCN [4], and SNIPER [38] are integrated into our framework. Note that aforementioned RFCN and SNIPER deploy Resnet-101, whereas YOLOv2 deploys Darknet-19 network. Similar to YADA framework with Faster RCNN baseline, we generate hard objects for these detectors separately and then train them with generated images. Table 6 shows results of YADA with these different detectors. On VOC 2007 test set, our YADA method further boosts YOLOv2, RFCN, and SNIPER baselines by 0.7%, 0.6%, and 0.3% in terms of mAP respectively. The results show that our method is also effective for modern object detectors and deep networks. Taking a closer look on the amount of improvement for each detector, YADA-YOLO gains the largest amount in mAP whereas YADA-SNIPER gain the smallest amount. This can be explained that we cannot teach much for the superior network as the inferior network.

5 Conclusions and future work

In this paper, we present a novel method, named YADA (You Always Dream Again), for generic object detection. The major contribution of this paper lies in two-fold: firstly, we apply lucid data synthesizing on the training set by mining the hard examples and cloning them to the same context locations. Different from previous data augmentation works, our synthesized data is generated with clear criteria. Secondly, we utilize a dual-level of deep

networks leveraged with the synthesized data. Our framework structure is designed to flexibly combine the two level of deep networks through a fusion scheme. The extensive experiments on two benchmarks, PASCAL VOC and KITTI, demonstrate the superiority of our approach over the state-of-the-art methods. In the future, we would like to investigate our work in more complicated deep networks for object detection. We also aim to apply the YADA philosophy to other computer vision tasks.

Acknowledgments This research is funded by Viet Nam National University Ho Chi Minh City (VNU-HCM) under Grant No. B2017-26-01.

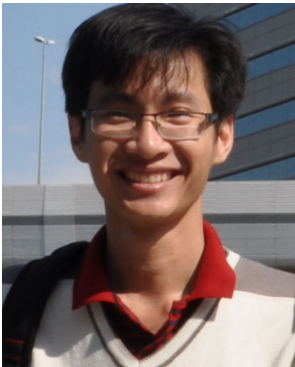
References

1. Cheng G, Zhou P, Han J (2016) Rfd-cnn: rotation-invariant and fisher discriminative convolutional neural networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2884–2893
2. Cheng G, Han J, Zhou P, Xu D (2019) Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection. *IEEE Trans Image Process* 28(1):265–278
3. Chu M, Wu S, Gu Y, Xu Y (2017) Rich features and precise localization with region proposal network for object detection. In: Chinese Conference on biometric recognition. Springer, pp 605–614
4. Dai J, Li Y, He K, Sun J (2016) R-fcn: object detection via region-based fully convolutional networks. In: *Advances in neural information processing systems*, pp 379–387
5. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: 2005 IEEE Computer society conference on computer vision and pattern recognition (CVPR 2005), pp 886–893
6. Dwibedi D, Misra I, Hebert M (2017) Cut, paste and learn: surprisingly easy synthesis for instance detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1301–1310
7. Everingham M, Gool LJV, Williams CKI, Winn JM, Zisserman A (2010) The pascal visual object classes (VOC) challenge. *Int J Comput Vis* 88(2):303–338
8. Everingham M, Eslami SA, Van Gool L, Williams CK, Winn J, Zisserman A (2015) The pascal visual object classes challenge: a retrospective. *Int J Comput Vis* 111(1):98–136
9. Felzenszwalb PF, McAllester DA, Ramanan D (2008) A discriminatively trained, multiscale, deformable part model. In: 2008 IEEE Computer society conference on computer vision and pattern recognition
10. Gaidon A, Wang Q, Cabon Y, Vig E (2016) Virtual worlds as proxy for multi-object tracking analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4340–4349
11. Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition (CVPR), pp 3354–3361
12. Girshick R (2015) Fast R-CNN. In: 2015 IEEE international conference on computer vision, pp 1440–1448
13. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on computer vision and pattern recognition, pp 580–587
14. Gupta S, Girshick R, Arbeláez P, Malik J (2014) Learning rich features from rgb-d images for object detection and segmentation. In: European conference on computer vision. Springer, pp 345–360
15. Han J, Zhang D, Cheng G, Liu N, Xu D (2018) Advanced deep-learning techniques for salient and category-specific object detection: a survey. *IEEE Signal Process Mag* 35(1):84–100
16. Handa A, Pătrăucean V, Stent S, Cipolla R (2016) Scenenet: an annotated model generator for indoor scene understanding. In: 2016 IEEE International conference on robotics and automation (ICRA). IEEE, pp 5737–5743
17. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
18. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: 2017 IEEE International conference on computer vision (ICCV). IEEE, pp 2980–2988
19. Huang G, Liu Z, Weinberger KQ, van der Maaten L (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 1, p 3
20. Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. arXiv:1602.07360

21. Johnson-Roberson M, Barto C, Mehta R, Sridhar SN, Rosaen K, Vasudevan R (2017) Driving in the matrix: can virtual worlds replace human-generated annotations for real world tasks? In: 2017 IEEE International conference on robotics and automation (ICRA). IEEE, pp 746–753
22. Kahan TL, LaBerge S (1994) Lucid dreaming as metacognition: implications for cognitive science. *Consciousness Cogn* 3(2):246–264
23. Khanh-Duy N, Khang N, Duy-Dinh L, Duc AD, Tam VN (2019) You always look again: Learning to detect the unseen objects. *J. Vis. Commun. Image Represent.* 60:206–216
24. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, pp 1097–1105
25. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, pp 1106–1114
26. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. [arXiv:1708.02002](https://arxiv.org/abs/1708.02002)
27. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: single shot multibox detector. In: 14th European conference on computer vision, ECCV 2016. Springer
28. Mahajan D, Girshick R, Ramanathan V, He K, Paluri M, Li Y, Bharambe A, van der Maaten L (2018) Exploring the limits of weakly supervised pretraining. [arXiv:1805.00932](https://arxiv.org/abs/1805.00932)
29. Peng X, Sun B, Ali K, Saenko K (2015) Learning deep object detectors from 3d models. In: *Proceedings of the IEEE international conference on computer vision*, pp 1278–1286
30. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. [arXiv preprint](https://arxiv.org/abs/1707.08397)
31. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. [arXiv:1804.02767](https://arxiv.org/abs/1804.02767)
32. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 779–788
33. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. In: *Proceedings of advances in neural information processing systems*, pp 91–99
34. Rolnick D, Tegmark M (2017) The power of deeper networks for expressing natural functions. [arXiv:1705.05502](https://arxiv.org/abs/1705.05502)
35. Ros G, Sellart L, Materzynska J, Vazquez D, Lopez AM (2016) The synthia dataset: a large collection of synthetic images for semantic segmentation of urban scenes. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 3234–3243
36. Shrivastava A, Gupta A, Girshick R (2016) Training region-based object detectors with online hard example mining. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 761–769
37. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. [arXiv:1409.1556](https://arxiv.org/abs/1409.1556)
38. Singh B, Najibi M, Davis LS (2018) Sniper: efficient multi-scale training. In: *Advances in neural information processing systems*, pp 9310–9320
39. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 1–9
40. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 2818–2826
41. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. In: *AAAI*, vol 4, p 12
42. Tam VN, Luoqi L, Khang N (2016) Exploiting generic multi-level convolutional neural networks for scene understanding. In: *ICARCV*, pp 1–6
43. Tam VN, Khanh N, Thanh-Toan D (2019) Semantic Prior Analysis for Salient Object Detection. *IEEE Trans. Image Processing* 28(6):3130–3141
44. Tam VN, Qi Z, Shuicheng Y (2018) Attentive Systems: A Survey. *Int. J. Comput. Vis.* 126(1):86–110
45. Tremblay J, Prakash A, Acuna D, Brophy M, Jampani V, Anil C, To T, Cameracci E, Boochoon S, Birchfield S (2018) Training deep networks with synthetic data: bridging the reality gap by domain randomization. [arXiv:1804.06516](https://arxiv.org/abs/1804.06516)
46. Van de Sande KE, Uijlings JR, Gevers T, Smeulders AW (2011) Segmentation as selective search for object recognition. In: 2011 IEEE International conference on computer vision (ICCV). IEEE, pp 1879–1886
47. Varol G, Romero J, Martin X, Mahmood N, Black MJ, Laptev I, Schmid C (2017) Learning from synthetic humans. In: 2017 IEEE Conference on computer vision and pattern recognition (CVPR 2017). IEEE, pp 4627–4635

48. Viola PA, Jones MJ (2004) Robust real-time face detection. *Int J Comput Vis* 57(2):137–154
49. Wang X, Yang M, Zhu S, Lin Y (2015) Regionlets for generic object detection. *IEEE Trans Pattern Anal Mach Intell* 37(10):2071–2084
50. Zhang D, Han J, Yang L, Xu D (2018) Spftn: a joint learning framework for localizing and segmenting objects in weakly labeled videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*
51. Zhang D, Han J, Zhao L, Meng D (2019) Leveraging prior-knowledge for weakly supervised object detection under a collaborative self-paced curriculum learning framework. *Int J Comput Vis* 127(4):363–380
52. Zhou Z-H, Feng J (2017) Deep forest: towards an alternative to deep neural networks: arXiv:[1702.08835](https://arxiv.org/abs/1702.08835)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Khanh-Duy Nguyen is a PhD student at University of Information Technology, Ho Chi Minh City, Vietnam. Prior to that, he obtained his B.S. and M.S. degrees from the University of Science, Ho Chi Minh City, Vietnam in 2007 and 2011, respectively. His research interests include computer vision, multimedia analysis and deep learning.



Khang Nguyen received his B.S degree and M.S degrees in Computer Science from University of Science, VNUHCM, Vietnam in 1990 and 1995. He received his Ph.D. degree in 2012 from the University of Science, VNUHCM, Vietnam. Currently, he is the Vice-President of University of Information Technology, VNUHCM, Vietnam. His research interests include artificial intelligence and computer vision.



Duy-Dinh Le is a scientist at University of Information Technology, Vietnam. He received his BS and MS degrees in 1995 and 2001, from the University of Science, Ho Chi Minh City, Vietnam, and his PhD degree in 2006 from The Graduate University for Advanced Studies (SOKENDAI), Japan. He was an associate professor at the National Institute of Informatics (NII), Japan from 2013 to 2016. His research interests include semantic concept detection, video analysis and indexing, pattern recognition, machine learning and data mining.



Duc Anh Duong is a professor at the University of Information Technology, Ho Chi Minh City, Vietnam. He obtained his B.Sc and M.Sc in computer science from the University of Ho Chi Minh City in 1990 and 1995, respectively. He received his Ph.D. degree in mathematics from the University of Science, VNUHCM, Vietnam in 2002. His research interests include image processing, computer vision and pattern recognition, cryptography and security, geographic information systems, and computer graphics. He was a visiting researcher at Japan Advanced Institute of Science and Technology (JAIST), Japan from 2008 to 2009. He is currently Chair of Program of Information Technology and Electronics of Ho Chi Minh City, and Chair of Program of Information Security of Ho Chi Minh City, Vietnam. He is a member of the ACM and IEEE.



Tam V. Nguyen is an Assistant Professor at Department of Computer Science, University of Dayton. He received PhD degree in National University of Singapore (NUS) in 2013. His research topics include computer vision, applied deep learning, multimedia content analysis, and mixed reality. He has authored and co-authored 60+ research papers with 1000+ citations according to Google Scholar. His works were published at IJCV, IEEE T-MM, IEEE T-CSVT, Neurocomputing, ECCV, IJCAI, AAAI, and ACM Multimedia. He is an IEEE Senior Member.