Noname manuscript No.
(will be inserted by the editor)

Learning Accurate Personal Protective Equipment Detection from Virtual Worlds

Marco Di Benedetto · Fabio Carrara · Fabrizio Falchi · Claudio Gennaro · Enrico Meloni · Giuseppe Amato

Received: date / Accepted: date

Abstract Deep learning has achieved impressive results in many machine learning tasks such as image recognition and computer vision. Its applicability to supervised problems is however constrained by the availability of high-quality training data consisting of large numbers of humans annotated examples (e.g. millions). To overcome this problem, recently, the AI world is increasingly exploiting artificially generated images or video sequences using realistic photo rendering engines such as those used in entertainment applications. In this way, large sets of training images can be easily created to train deep learning algorithms. In this paper, we generated photo-realistic synthetic image sets to train deep learning models to recognize the correct use of personal safety equipment (e.g., worker safety helmets, high visibility vests, ear protection devices) during at-risk work activities. Then, we performed the adaptation of the domain to real-world images using a very small set of real-world images. We demonstrated that training with the synthetic training set generated and the use of the domain adaptation phase is an effective solution for applications where no training set is available.

Keywords Deep Learning · Virtual Dataset · Transfer Learning · Domain Adaptation · Detection · Personal Protective Equipment

1 Introduction

It is estimated that every day around six thousand people die in the world due to accidents at work or occupational diseases, causing more than 2.3 million deaths a year. Many of these accidents could be prevented by the simple use of personal safety equipment, such as helmets or reflective vests. However, it is not always possible to effectively control whether such equipment is actually

{Authors}
Institute of Information Science and Technologies, National Research Council, Italy, E-mail: {name.surname}@isti.cnr.it

used. Artificial Intelligence (AI) can be of great help by constantly analyzing the working environment with a camera and warning workers who do not comply with the rules.

To this end, the AI sector known as supervised machine learning has achieved significant success in a variety of application domains. These achievements have been so impressive that they have attracted increasing attention from the scientific community to the production of annotated datasets with which to train learning algorithms. In the era of big data, the availability of examples such as images or videos is not considered a problem. However, these data must be annotated by humans before they can be used, e.g. by adding class labels or visual masks, and in many specific application domains, this can be very expensive or even impossible.

Indeed, although a large amount of annotated data is already available and successfully used to produce important academic results and commercially viable products, there is still a huge amount of scenarios where laborious human intervention is required to produce high-quality training sets. These scenarios include, but are not limited to, the detection of safety equipment, self-driving cars, the detection of firearms.

To address this problem and make up for the lack of annotated examples in a variety of scenarios, the research community has begun to increasingly leverage the use of programmable virtual scenarios to generate synthetic visual data sets as well as associated annotations. For example, in image-based deep learning techniques, the use of a modern rendering engine (i.e. capable of producing photo-realistic images) has proven to be a valuable tool for the automatic generation of large data sets (see Section 2). The advantages of this approach are remarkable. In addition to making up for the lack of data sets in some particular application domains, these synthetic datasets do not create problems with existing laws about the privacy of individuals related to facial detection, such as the European of the General Data Protection Regulation (GDPR).


a)


b)

Fig. 1 Examples of safety equipment: *a)* real photograph of worker wearing helmets and high-visibility vests, and *b)* virtual rendering with people with helmets, welding mask, ear protections, and high-visibility vests.

In this paper, we investigate the effectiveness of rendering engines in generating realistic image sequences to train machine learning algorithms to address

the problem of detection and recognition in scenarios where no or insufficient annotated data is available. In particular, the study focuses on the context of visual detection of safety equipment (see Figure 1), for which, to the best of our knowledge, no public dataset exists.

To this end, we show how a known deep neural network exploiting the *transfer learning* approach can achieve cutting-edge results in object detection tasks when trained with virtually generated images containing people equipped with safety items (such as high visibility jackets and helmets) and then adapted to the real world domain using some training examples retrieved from the Internet. More in detail, we contribute to this field with the following results:

- automatic generation of a virtual training set for the recognition of personal security equipment, with different scene conditions,
- provide an annotated test set of real-world images, and
- competitive results with state of the art detectors tested for such scenarios.

We will see that, on the very few real-world examples available, the use of virtual images dramatically increases system performance in terms of accuracy. The dataset that we created is made publicly available to the research community [1].

This work extends the paper we presented at CBMI 2019 that received the best paper award [2]. The main extension is on experimenting our approach not only on the YOLO architecture but also on Faster-RCNN, another commonly used object detection method. By doing this, we did not only achieved better performance, bu we demonstrated that the overall approach was not specific to the YOLO architecture.

The rest of the paper is organized as follow: Section 2 gives an overview of existing methods based on virtual environments; Section 3 describes how we used an existing rendering engine and the policy to create the dataset and the test set; Section 4 discusses our detection method; Section 5 shows our experimental results; finally Section 6 concludes.

2 Related Work

With the advent of deep learning, object detection technologies have achieved accuracies that were unimaginable only a few years ago. YOLO architectures [3, 4] and Faster-RCNN [5] are today de facto-standard architectures for the object detection task. They are trained on huge generic annotated datasets, such as ImageNet [6], MS COCO [7], Pascal [8] or OpenImages v4 [9]. These datasets collect an enormous amount of pictures usually taken from the web and they are *manually* annotated.

With the need for huge amounts of labeled data, virtually generated datasets have recently gained great interest. The possibility of learning features from virtual data and validating them on real scenarios was explored in [10]. Unlike our work, however, they did not explore deep learning approaches. In [11], computer-generated imagery was used to study trained CNNs to qualitatively

and quantitatively analyze deep features by varying the network stimuli according to factors of interest, such as to object style, viewpoint, and color. The works [12,13] exploit the popular Unreal Engine 4 (UE4) to build virtual worlds and use them to train and test deep learning algorithms.

The problem of transferring deep neural network models trained in simulated virtual worlds to the real world for vision-based robotic control was explored in [14]. In a similar scenario, [15] developed an end-to-end active tracker trained in a virtual environment that can adapt to real-world robot settings. To handle the variability in real-world data, [16] relied upon the technique of domain randomization, in which the parameters of the simulator—such as lighting, pose, object textures were randomized in non-realistic ways to force the neural network to learn the essential features of the object of interest. A deep learning model was trained in [17] to drive in a simulated environment and adapted it for the visual variation experienced in the real world.

[18,19] focused their attention on the possibility to perform domain adaptation in order to map virtual features onto real ones. Richter et al. [20] explored the use of the video game *Grand Theft Auto V* (GTA-V) [21] for creating large-scale pixel-accurate ground truth data for training semantic segmentation systems. In [22], they used GTA-V for training a self-driving car and generated around 480,000 images for training. This work evidenced how GTA-V can indeed be used to automatically generate a large dataset. The use of GTA-V to train a self-driving car was explored also in [23], where images from the game were used to train a classifier for recognizing the presence of stop signs in an image and estimate their distance. In [24] a different game was used for training a self-driving car: TORCS, an open-source racing simulator with a graphics engine less focused on realism than GTA-V.

Authors in [25] created a dataset taking images from GTA-V and demonstrated that it is possible to reach excellent results on tasks such as real people tracking and pose estimation.

[26] also used GTA-V as the virtual world but, unlike our method, they used Faster-RCNN and they concentrated on vehicle detection validating their results on the KITTI dataset. Instead, [27] used a synthetically generated virtual dataset to train a simple convolutional network to detect objects belonging to various classes in a video.

3 Training Set from Virtual Worlds

In this paper, we show that a low cost and off-the-shelf virtual rendering environment represents a viable solution for generating a high-quality training set for scenarios lacking enough real training data. This method allows generating a very large amount of annotated images, with the possibility of scenery changes like location, contents, and even weather conditions, with very little human intervention.

In this work, we used the generated training set to train a *You Only Look Once* (YOLO) neural system [3, 4] for its efficiency, and a modification of a

Faster-RCNN [5] for its high detection accuracy (see Section 4). However, the applied methodology can be used in other machine learning tools.

We used the *Rockstar Advanced Game Engine* (RAGE) from the GTA-V computer game, and its scripting ability to deploy a series of pedestrians with and without safety equipment in different locations of the game map. The *RAGE Plugin Hook* [28] allowed us to create and inject our C# scripts into the game.

Our scripts use the plugin API to add pedestrians with chosen equipment in various locations of the game map, place cameras in places where we want to take pictures, check that objects are in the field of view and not occluded, recover 3D meshes bounding boxes from the rendering engine, and save game screenshots (i.e., our dataset images) and their associated annotations (bounding boxes and classes).

Personal safety equipment that we consider includes, for example, high-visibility vests, helmets, welding masks, and others. In addition to persons wearing these types of equipment, we also generate pedestrians without protections, where we annotate, person, bare head, bare chest (see Figure 2 images as an example).

The generation of the virtual dataset required first to configure the RAGE engine to create various types of scenarios (Section 3.1). Then, the RAGE engine was used to capture images along with annotations. For every image, the annotations (coordinates of the bounding boxes and identities of relevant elements) were retrieved from the RAGE engine through our script (Section 3.2). We used this approach both for creating the virtual world training set and the virtual world validation set. The dataset was eventually completed by adding a real-world test set, composed of real-world images, to test the accuracy of the trained neural network on real scenes (Section 3.3).

3.1 Scenario Creation

To generate the training scenario we used the plugin API to customize the following game features:

- **Camera**: used to set up the viewpoints from which the scenario must be recorded.
- **Pedestrians**: used to set up the number of people in the scene and their behavior, chosen from the set offered by the game engine, such as wandering around an area, chatting between themselves, fighting, and so on.
- **Place**: used to set up the place where the pedestrians will be generated; there is a series of game map preset places, plus user-defined locations identified by map coordinates.
- **Time**: used to set up the time of day during which the scene takes place.
- **Weather**: used to set up the weather conditions during the animation.

We used nine different game map locations with three different weather conditions each to create the virtual training set. From these, we acquired a total

Fig. 2 Detected objects in a working scenario. The real scene depicted in each column is analyzed using three different version of the trained networks: YOLO trained with COCO and fine-tuned with real images (YCR, first row), YOLO trained with COCO and fine tuned with virtual renderings and real images (YCVR, second row), and Faster RCNN trained with COCO and fine-tuned with virtual renderings and real images (YCVR, third row). Note that difference upon networks and datasets pays in both detection accuracy (rightmost more precise), and run-time resources (see end of Section 5.2).

of 126,900 images with an average of 12 persons per shot. The virtual validation set spans one location with three weather conditions, and consists of 350 images with an average of 12 persons each. Therefore, in the end, we have 30 different scenarios where virtual world images were extracted from.

3.2 Dataset Annotation

Dataset annotation is the process which creates the annotated images for the dataset. In our case, we annotate the following elements (see Figure 3):

- **Head**: a bare head (without protection devices)
- **Helmet**: a head wearing a helmet
- **Welding Mask**: a head wearing a welding mask
- **Ear Protection**: a head wearing hearing protection
- **Person**: a full-body person
- **Chest**: the bare chest (without protection vests)

Fig. 3 Examples of safety equipment objects detected by our system.

– **High-Visibility Vest**: a chest with a high visibility vest (HVV)

For each viewpoint setup in the scenario, we process every object to extract its position on the 2D image. This is done by first calculating the geometry of its transformed 3D bounding box, then approximately testing the box visibility, and finally extracting the image 2D bounding box by contouring the 3D box vertices (see Figure 4). The visibility is checked by testing the occlusion of line-of-sight rays from the camera to a certain fixed amount of point in the box volume, and the object is considered visible if at least one ray is not occluded.

3.3 Real World Test Set

The motivation of this work is to prove that it is possible to train a system with a virtual world even when it is supposed to be used in the real-world. To test the performance of the trained neural network in the real-world, we created a real-world test set using copyright-free photographs of people wearing safety equipment. The set is composed of 180 images (see Figure 5) showing persons with and without the items listed in Section 3.2, each associated with manually created annotations of bounding boxes and element identities.

4 Method

The backbone of our detection algorithm is based on deep convolutional neural networks able to detect, in a single image, the objects which they have been trained for. The detection ends with a list of 2D bounding boxes, each associated with a class label referring to the recognized object. In our implementation, we experimented with two different detection networks, i.e. the YOLO *v3* [4] (hereafter abbreviated with YOLO) network with the *Darknet-53* in its core, and the improved version of Faster-RCNN [5] network that

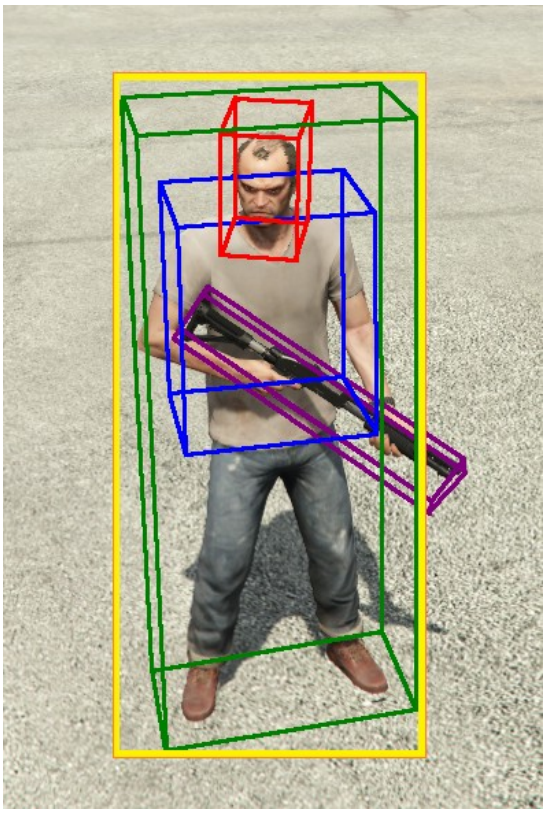

Fig. 4 Bounding box estimation: oversized approximation with respect to on-screen projections. With the available API, the hooked virtual engine can provide the bounding boxes of individual 3D meshes, overestimated due to collision proxy expansion for animations. Not being able to access the original 3D geometry and the current animation frame, a working strategy is to project on-screen the eight corners of the 3D bounding box, and then take their containing minimum rectangle as an approximate annotation. In the image, the green 3D box is annotated with the yellow 2D rectangle.

Fig. 5 Real-World Validation Set: composed of 180 copyright-free images, our validation set is available at the project website.

includes a Feature Pyramid Network [29] and the ResNet-50 as backbone. The chosen models are representative of the two major architectures in object detection: the former belongs to the family of single-stage detectors that are fast and produce dense detections, while the latter belongs to two-stage detectors, usually slower but more accurate systems that first locate candidate regions and then provide predictions for them. We trained both to recognize personal safety equipment components, as shown in the following.

4.1 Transfer Learning

As anticipated, we use the generated virtual world training set to train the detection networks to detect and recognize our elements of interest in images. In particular, we adapt the detection networks to our scenario using *transfer learning*. Our hypothesis is that a pre-trained network already embeds enough knowledge that allows us to specialize it in a new scenario, leveraging on the transfer learning capability of deep neural networks and on training sets generated from a virtual world.

The purpose of transfer learning is to exploit the knowledge stored in the network as a starting point to extend the detection capability to the new set of objects. With a trained deep convolutional neural network, its first layers have learned to identify features that are more and more complex according to layer depth; for example, the first layer will be able to detect straight borders, the second layer smooth contours, the third some kind of color gradients, and so on while arriving at last layers capable of identifying entire objects.

In our case, we used a detection network pre-trained on the COCO dataset. Concerning YOLO, we fine-tuned it by blocking the learning parameters update of the first part of the network, and allowing updates only in the last sections. Specifically, we kept the first 81 (i.e., the feature extractors) of the total 106 layers, and froze the weights of the first 74. The network was trained for 24,000 iterations, that is 11 epochs with the following parameters: batch size 64, decay 0.0005, learning rate 0.001, momentum 0.9, IoU threshold: 0.5, confidence threshold: 0.25.

For the Faster-RCNN network, we fine-tuned the entire network on the new objects allowing updates in every trainable part of the detector. Being a significantly bigger network with respect to YOLO, we trained with a smaller batch size of two and kept the same values for the other parameters. Interestingly, only two epochs were sufficient for Faster-RCNN to converge on our virtual dataset.

As explained in Section 3, our virtual dataset is composed of 30 scenarios, 27 of which were used as the training set and three were left for validation. The three scenarios of the validation set contain 13,500 images. From these, 350 images were randomly selected to form the virtual validation dataset. In this way, a new set of objects are recognized by the network.

4.2 Evaluation Metrics

To evaluate the performance of our implementation, we applied the standard measures used in the object detection literature, i.e., *Intersection over Union* (IoU) based on the area of the detected (D) and real (V) bounding boxes, and *Precision* (Pr) and *Recall* (Rc) based on true (T) / false (F) positive (P) / negative (N) detections:

- IoU $= (D \cap V)/(D \cup V)$
- Pr $= TP/(TP + FP)$
- Rc $= TP/(TP + FN)$

Detected bounding boxes are associated with a confidence score, ranging from 0 to 1, and are considered in the output if and only if their confidence score is greater than a configurable threshold. Given the above definitions, we calculate the *mean Average Precision* (mAP) as the average of the maximum precision at different recall values.

5 Experiments and Results

We conducted a series of experiments with the two neural networks on the virtual and real datasets, as explained hereafter.

5.1 Experimental Setups

We trained and evaluated three variations of the two networks on both virtual and real images: YCV, which is YOLO base trained on COCO and fine-tuned with Virtual data; YCVR, which is YCV additionally fine-tuned with Real data; YCR, which is YOLO base trained on COCO and fine-tuned with Real data. Following the same rationale, we obtain the same configurations for Faster-RCNN, that are FCV, FCVR, FCR.

To obtain YCVR and FCVR, we split the real-world dataset in to two parts with 100 images each: a training part and a testing part. We used the training part to apply domain adaptation from virtual to real on the YCV and FCV networks by performing fine-tuning. To choose the set of weights from which to start, we validated each of them on the training part, choosing the one with the highest mAP.

To better evaluate the benefit contributed by the virtual world training set, we also fine-tuned the base networks , pre-trained on COCO, with the same 100 real images used for obtaining YCR and FCR.

5.2 Results

Results are reported in Table 1. YCV and FCV obtain respectively 87.2% and 95.0% mAP when tested on virtual images, and when tested on real-world

images, they obtain respectively 55.1% mAP and 42.6%. Most of the AP loss is caused by the classes Head, Welding Mask, Ear Protection, and Chest. We believe that this is because in real life there are many more variations of these object classes than those the game can render.

We want to note that, on virtual world testing, both YCV and FCV obtain better mAP after more iterations, while on real-world testing better performance is reached before in the training phase. This implies that the best performing set of weights for the virtual world test is not the best also for real-world validation, as further training seems to induce a bias in the network towards unique peculiarities of the source domain.

Table 1 *mAP* comparison of our networks on Virtual validation or Real testing

Network	Test	Head	Helmet	Weld. Mask	Ear Prot.	Chest	HVV	Person	mAP
YCV	V	89.7%	86.7%	75.5%	89.0%	89.7%	90.0%	89.7%	87.2%
YCV	R	36.3%	74.1%	27.3%	55.6%	45.7%	69.9%	76.9%	55.1%
YCR	R	44.1%	52.2%	42.3%	62.0%	59.1%	60.7%	80.6%	57.3%
YCVR	R	78.8%	73.3%	66.3%	74.0%	74.7%	78.6%	87.1%	76.1%
FCV	V	95.2%	98.1%	84.7%	97.9%	95.6%	98.1%	95.2%	95.0%
FCV	R	33.7%	64.9%	5.1%	26.9%	38.1%	63.3%	66.2%	42.6%
FCR	R	68.8%	76.5%	66.5%	65.7%	72.6%	74.3%	92.1%	73.8%
FCVR	R	73.6%	79.2%	69.6%	72.7%	76.5%	81.5%	86.6%	77.1%

YCVR obtains a significant boost and reaches 76.1 mAP. This means that fine-tuning with only 100 real images is very effective on a network that was previously fine-tuned with several similar virtual images. We also note that testing YCVR on the virtual world yields a lower mAP with respect to YCV. The main drop of AP, in this case, is seen on Head and Welding Mask, which are the classes with most differences between real and virtual.

YCR obtain 57.3 mAP when tested on the real-world test set. This result is just slightly better than that obtained by YCV, and by far worse than YCVR. This means that, to train the network for the new scenario, the contribution given by the virtual world training set is very relevant, and just a fine-tuning with a few images is enough to adapt the network back to the real-world domain.

Concerning Faster-RCNN, we observe the same trend even if the effects are smaller. Its architectural details enable the network to learn more effectively even in small data regimes, as suggested by FCV tested on the virtual test set (95.0% mAP) and FCR tested on the real set (73.8% mAP). However, this network shows a lower transferability with respect to domain change, as pointed out by FCV tested on the real dataset that obtains at peak 42.6% mAP. When domain adaptation by fine-tuning is applied (FCVR), we still obtain an improvement with respect to FCR, even if the boost is smaller than the one obtained by YOLO. We deem the inertia of Faster-RCNN to transferability of representations and its tendency to overfit are to believe the

main causes for this effect, and we think this could be mitigated using more clever domain adaptation techniques.

By inspecting the fine-tuned weights, we observed that lower-level filters do not change that much between models trained with virtual and real data, while mostly the higher-level network heads were responsible for the improved performance. This is somewhat expected, as we fine-tune models pretrained on COCO that already show good configurations of weights for low- and mid-level layers. We leave for future work a more in-depth analysis concerning feature attribution and visualization to pinpoint more precisely the differences on a representation level. On a practical standpoint, we observe two main positive effects brought by virtual data, that are: a) tighter box predictions, due to accurate annotations provided by the engine, and b) improvements on high-variability classes (e.g. head, chest), occluded objects, and corner cases (e.g. crouched people) for which the network can provide better higher-level representations leveraging all the variability that the engine can provide. Figure 2 shows some examples of these effects on samples from the real-world test set.

From a run-time perspective, we obtained a forward pass speed of 2.6 FPS with Faster RCNN and 6.4 FPS with YOLO, including the whole process (e.g., disk fetch, data submission to the GPU, and result read-back): given the measured performances and the application requirements, we can conclude that both implementations are ready to be implemented in real-time detection on real video systems without modifications.

6 Conclusions

Training deep neural networks in virtual environments has been recently proven to be of help when the number of available training examples for the specific task is low. In this work, we considered the task of learning to detect proper equipment in risky human activity scenarios.

We created and made available two datasets: the first one has been generated using a virtual reality engine (RAGE from GTA-V); the second one is composed of real photos.

In our experiments, we trained object detectors based on deep convolutional networks on the virtual dataset and tested on the real images as well as using just a small number of real photos to fine-tune the deep neural network we trained in the virtual environment. The experiments we conducted demonstrated that training on virtual world images, and executing a step of domain adaptation with a limited number of real images, is effective. Obtained performance when training with virtual world images and adapting to the domain with a few real images is higher than just fine-tuning an existing network with a few real images for the scenario at hand. We plan to use the same virtual environment to train to detect people using weapons (see Figure 4), and adopt state-of-the-art domain adaptation techniques (e.g., using transferable features as in [30]) to better close the gap between virtual a real worlds.

References

1. Enrico Meloni, Marco Di Benedetto, Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, "Project Website," http://aimir.isti.cnr.it/vw-ppe, 2019.
2. M. di Benedetto, E. Meloni, G. Amato, F. Falchi, and C. Gennaro, "Learning safety equipment detection using virtual worlds," in *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*, Sep. 2019, pp. 8–13.
3. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
4. J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.
5. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.
6. J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.
7. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.
8. M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan 2015.
9. A. Kuznetsova, H. Rom, N. Alldrin, J. R. R. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari, "The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale," *CoRR*, vol. abs/1811.00982, 2018.
10. J. Marín, D. Vázquez, D. Gerónimo, and A. M. López, "Learning appearance in virtual scenarios for pedestrian detection," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 137–144.
11. M. Aubry and B. C. Russell, "Understanding deep features with computer-generated imagery," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2875–2883.
12. W. Qiu and A. Yuille, "Unrealcv: Connecting computer vision to unreal engine," in *European Conference on Computer Vision*. Springer, 2016, pp. 909–916.
13. K.-T. Lai, C.-C. Lin, C.-Y. Kang, M.-E. Liao, and M.-S. Chen, "Vivid: Virtual environment for visual deep learning," in *Proceedings of the 26th ACM International Conference on Multimedia*, ser. MM '18. New York, NY, USA: ACM, 2018, pp. 1356–1359.
14. Z.-W. Hong, C. Yu-Ming, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, H.-K. Yang, B. H.-L. Ho, C.-C. Tu, Y.-C. Chang, T.-C. Hsiao *et al.*, "Virtual-to-real: Learning to control in visual semantic segmentation," *arXiv preprint arXiv:1802.00285*, 2018.
15. W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang, "End-to-end active object tracking and its real-world deployment via reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, 2019.
16. J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 969–977.
17. A. Bewley, J. Rigley, Y. Liu, J. Hawke, R. Shen, V.-D. Lam, and A. Kendall, "Learning to drive from simulation without real world labels," *arXiv preprint arXiv:1812.03823*, 2018.
18. D. Vázquez, A. M. Lopez, and D. Ponsa, "Unsupervised domain adaptation of virtual and real worlds for pedestrian detection," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Nov 2012, pp. 3492–3495.

19. D. Vázquez, A. M. López, J. Marín, D. Ponsa, and D. Gerónimo, "Virtual and real world adaptation for pedestrian detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 797–809, April 2014.
20. S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision*. Springer, 2016, pp. 102–118.
21. Rockstar Games, Inc., "Grand Theft Auto - V," https://www.rockstargames.com/V, 2013.
22. M. Martinez, C. Sitawarin, K. Finch, L. Meincke, A. Yablonski, and A. Kornhauser, "Beyond grand theft auto v for training, testing and enhancing deep learning in self driving cars," *arXiv preprint arXiv:1712.01397*, 2017.
23. A. Filipowicz, J. Liu, and A. Kornhauser, "Learning to recognize distance to stop signs using the virtual world of grand theft auto 5," Tech. Rep., 2017.
24. C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.
25. M. Fabbri, F. Lanzi, S. Calderara, A. Palazzi, R. Vezzani, and R. Cucchiara, "Learning to detect and track visible and occluded body joints in a virtual world," in *European Conference on Computer Vision (ECCV)*, 2018.
26. M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" *CoRR*, vol. abs/1610.01983, 2016.
27. E. Bochinski, V. Eiselein, and T. Sikora, "Training a convolutional neural network for multi-class object detection using solely virtual world data," in *Advanced Video and Signal Based Surveillance (AVSS), 2016 13th IEEE International Conference on*. IEEE, 2016, pp. 278–285.
28. "RAGE Plugin Hook," https://ragepluginhook.net, 2013.
29. T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.
30. M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 97–105.

Acknowledgements This work was partially supported by "Automatic Data and documents Analysis to enhance human-based processes" (ADA), funded by CUP CIPE D55F17000290009, and by the AI4EU project, funded by EC (H2020 - Contract n. 825619). We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Jetson TX2 board used for this research.