electronics

MDPI

Article

Improving Deep Object Detection Algorithms for Game Scenes

Minji Jung [1], Heekyung Yang [2,*,†] and Kyungha Min [3,*,†]

[1] Department of Human Centric Intelligent Information, Sangmyung University, Seoul 03016, Korea; mjjung@smu.ac.kr
[2] Division of SW Convergence, Sangmyung University, Seoul 03016, Korea
[3] Department of Computer Science, Sangmyung University, Seoul 03016, Korea
* Correspondence: yanghk@smu.ac.kr (H.Y.); minkh@smu.ac.kr (K.M.); Tel.: +82-2-781-7170 (H.Y.); +82-2-2287-5377 (K.M.)
† These authors contributed equally to this work.

Abstract: The advancement and popularity of computer games make game scene analysis one of the most interesting research topics in the computer vision society. Among the various computer vision techniques, we employ object detection algorithms for the analysis, since they can both recognize and localize objects in a scene. However, applying the existing object detection algorithms for analyzing game scenes does not guarantee a desired performance, since the algorithms are trained using datasets collected from the real world. In order to achieve a desired performance for analyzing game scenes, we built a dataset by collecting game scenes and retrained the object detection algorithms pre-trained with the datasets from the real world. We selected five object detection algorithms, namely YOLOv3, Faster R-CNN, SSD, FPN and EfficientDet, and eight games from various game genres including first-person shooting, role-playing, sports, and driving. PascalVOC and MS COCO were employed for the pre-training of the object detection algorithms. We proved the improvement in the performance that comes from our strategy in two aspects: recognition and localization. The improvement in recognition performance was measured using mean average precision (mAP) and the improvement in localization using intersection over union (IoU).

Keywords: object detection; game scene; deep learning; YOLO; SSD; R-CNN; EfficientDet

Citation: Jung, M.; Yang, H.; Min, K. Improving Deep Object Detection Algorithms for Game Scenes. *Electronics* **2021**, *10*, 2527. https://doi.org/10.3390/electronics10202527

Academic Editor: Xenophon Zabulis

Received: 29 August 2021
Accepted: 8 October 2021
Published: 17 October 2021

1. Introduction

Computer games have been one of the most popular applications for all generations since the dawn of the computing age. Recent progress in computer hardware and software has presented computer games of high quality. Nowadays, e-sports, playing or watching computer games, have become some of the most popular sports. E-sports are newly emerging sports where professional players compete in highly popular games, such as Starcraft and League of Legends (LoL), while millions of people watch them. Consequently, e-sports have become one of the most popular types of content on various media channels, including YouTube and Tiktok. From these trends, analyzing game scenes by recognizing and localizing objects in the scenes has become an interesting research topic.

Among the many computer vision algorithms including object recognition and object detection, localization and segmentation are candidates for analyzing game scenes. In analyzing game scenes, both recognizing and localizing objects in the scene are required. Therefore, we select object detection algorithms for analyzing game scenes. Object detection algorithms can identify thousands of objects and draw bounding boxes for objects in real-time. At this point, we have a question in relation to applying object detection algorithms to game scenes: "Can the object detection algorithms trained by real scenes be applied to game scenes?"

Detecting objects in game scenes is not a straightforward problem that can be resolved by applying existing object detection algorithms. The recent progress in computing hardware and software techniques presents diverse visually pleasing rendering styles to

computer games. Some games are rendered in a photorealistic style, while some are in a cartoon style. Furthermore, various depictions of a game scene with various colors and tones present a distinctive game scene style. Some cartoon-based games present their deformed characters and objects according to their original cartoons. Therefore, detecting various objects in diverse games can be challenging.

Existing deep-learning-based object detection algorithms show satisfactory detection performance for images captured from the real world. We selected five of the most widely-used deep object detection algorithms: YOLOv3 [1], Faster R-CNN [2], SSD [3], FPN [4] and EfficientDet [5]. We also prepared two frequently used datasets, PascalVOC [6,7] and MS COCO [8], for training the object detection algorithms. We examined these algorithms in recognizing objects in game scenes.

We aimed to improve the performance of object recognition of these algorithms by retraining them using game scenes. We prepared eight games including various genres, such as first-person shooting, racing, sports, and role-playing. Two of the selected games presented cartoon-styled scenes. We excluded games with non-real objects. In many fantasy games, for example, dragons, orcs, and non-existent characters appear. We excluded these games since existing object detection algorithms are not trained to detect dragons or orcs.

We also tested a data augmentation scheme that produces cartoon-styled images for the images in frequently used datasets. Several widely used image abstraction and cartoon-styled rendering algorithms were employed for the augmentation process. We retrained the algorithms using the augmented images and measured their performances.

To prove that the performance of the object detection algorithms was improved using game scene datasets, we compared the comparison for two cases. One case was to compare PascalVOC and PascalVOC with game scenes, and the other case was to compare MS COCO and MS COCO with game scenes. For each case of comparisons, the five object detection algorithms were pre-trained with a frequently used dataset. After measuring the performance, we retrained the algorithms with game scenes and measured the performance. These performances were compared to prove our hypothesis that the object detection algorithms trained with the public dataset and game scenes showed better performance than the algorithms trained only with the public dataset.

We compared the pre-trained and retrained algorithms in terms of two metrics: mean average precision (mAP) and intersection over union (IoU). We examined the accuracy of recognizing objects with mAP and the accuracy of localizing objects with IoU. From this comparison, we could determine whether the existing object detection algorithms could be used for game scenes. Furthermore, we could also determine whether the object detection algorithms retrained with game scenes showed a significant difference from the pre-trained object detection algorithms.

The contributions of this study are summarized as follows:

- We built a dataset of game scenes collected from eight games.
- We presented a framework for improving the performance of object detection algorithms on game scenes by retraining them using game scene datasets.
- We tested whether the augmented images using image abstraction and stylization schemes can improve the performance of the object detection algorithms on game scenes.

This study is organized as follows. Section 2 briefly explains deep-learning-based object detection algorithms and presents several works on object detection techniques in computer games. We elaborate on how we selected object detection algorithms and games in Section 3. In Section 4, we explain how we trained the algorithms and present the resulting figures. In Section 5, we analyze the results and answer our RQ. Finally, we conclude and suggest future directions in Section 6.

2. Related Work

2.1. Deep Object Detection Approaches

Object detection, which extracts a bounding box around a target object from a scene, is one of the most popular research topics in computer vision. Many object detection

algorithms have been presented after the emergence of the histogram of the oriented gradient (HoG) algorithm [9]. Recently, the progress of deep learning techniques has accelerated object detection algorithms on a great scale. Many recent works, including the you only look once (YOLO) series [1,10,11], the region with a convolutional neural network (R-CNN) series [2,12,13], spatial pyramid pooling (SPP) [14] and the single-shot multibox detector (SSD) [3], have demonstrated impressive results in detecting diverse objects from various scenes.

YOLO detects objects by decomposing an image into $S \times S$ grid cells. We estimate B bounding boxes at each cell, each of which possesses a box confidence score and C conditional class probabilities. The class confidence score, which estimates the probability of an object belonging to a class in the cell, is computed by multiplying the box confidence score with the conditional class probability. YOLO is a CNN that estimates the class confidence score for each cell. Although YOLOv1 [10] has a very fast computational speed, it suffers from relatively low mAP and limited classes for detection. Redmon et al. later presented YOLOv2, also known as YOLO9000, which detects 9000 objects with improved precision [11]. They further improved YOLOv2's performance in YOLOv3 [1].

R-CNN, which is another mainstream deep object detection algorithm, employs a two-pass approach [12]. The first pass extracts a candidate region, where an object should go through a selective search and a region proposal network. In the second, they recognize the object and localize it using a convolutional network. Girshick presented fast R-CNN, improving computational efficiency [13], and Ren et al. presented faster R-CNN [2].

The SPP algorithm allows arbitrary size input for object detection [14]. It does not crop or warp input images to avoid distortion of the result. It devises an SPP layer before the fully connected (FC) layer to fix the size of feature vectors extracted from the convolution layers. SSD addresses the problem of YOLO, which neglects objects smaller than the grid [3]. The SSD algorithm applies an object detection algorithm to each feature map extracted through a series of convolutional layers. The detected information is merged into a final detection result by executing a fast non-maximum suppression.

FPN builds a pyramid structure on the images by reducing their resolutions [4]. FPN extracts features in a top-down approach and merges the extracted features in both high-resolution images and low-resolution images. In the high-resolution images, the features in low-resolution images are employed to predict the features in high-resolution images. The pyramid structure of FPN extracts more semantics on the features in low-resolution images. Therefore, FPN extracts features from the input image in a convincing way.

2.2. Object Detection in a Game

Utsumi et al. [15] presented a classical object detection and tracking method for a soccer game in the early days. They employed a color rarity and local edge property for their object detection scheme. They extracted objects with high edges from a roughly single-colored background. Compared to a real soccer game scene, their model shows a comparatively high detection rate.

Many researchers have applied the recent progress of deep-learning-based object detection algorithms to individual games.

Chen and Yi [16] presented a deep Q-learning approach for detecting objects in 30 classes from the classic game Super Smash Bros. They proposed a single-frame 19-layered CNN model, with five convolution layers, three pooling layers and three FC layers. Their model recorded 80% top-1 and 96% top-3 accuracies.

Sundareson [17] chose a specific data flow for in-game object classification. Their model also aimed to detect objects in virtual reality (VR). They converted 4K input images into 256×256 resolution for efficiency. Their model's performance exhibits very competitive results in implementing in-game and in-VR object classification using CUDA.

Venkatesh [18] surveyed and proposed SmashNet, a CNN-based object tracking scheme in games. This model recorded 68.25% classification precision for four characters

in fighting games by employing very effective structures. The author also developed KirbuBot, which performs basic commands on the positions of two tracked characters.

Liu et al. [19] employed faster R-CNN to implement the vision system of a game robot. They extracted features and position label mapping for a single object using ResNet100. The information about object movement in a game is tracked using the robot's camera to improve the accuracy and speed of the model recognition.

Chen et al. [20] attempted to address the multi-object tracking problem, which is crucial in video analysis, surveillance and robot control, using a deep-learning-based tracking method. They applied their method to a football video to demonstrate the performance of their method.

Tolmacheva et al. [21] used a YOLOv2 model to track a puck in an air hockey game. The air hockey game is an arcade game played by two players who aim to push a puck into the opponent's goal by moving a small hand-held stick. Since the puck moves with great velocity, exact detecting and tracking of an object is a challenging problem. They collected and prepared datasets from game images to predict the trajectory of a fixed object. Using YOLOv2 in a C implementation, they recorded 80% detection accuracy.

Yao et al. [22] presented a two-stage algorithm to detect and recognize "hero" characters from a video game named Honor of Kings. They applied a template-matching method to detect all heroes in the frames and devised a deep convolutional network to recognize the name of each detected hero. They employed InceptionNet and recorded a 99.6% F1 score with less than 5 ms recognition time.

Spijkerman and van der Harr [23] presented a vehicle recognition scheme for Formula One game scenes using several object detection algorithms, including HoG, support vector machine and faster R-CNN. They trained their models using images captured from the F1 2019 video game. Their models' precision and recall scores based on R-CNN, the best among the three models, record 97% and 99%, respectively. They applied the trained R-CNN model to real objects and achieved 93% precision.

Kim et al. [24] improved the performance of a safety zone monitoring system using game-engine-based internal traffic control plans (ITCPs). They used a deep-learning-based object detection algorithm to recognize and detect workers and types of equipment from aerial images. They also monitored unsafe activities of works by observing four rules. Through this approach, they emphasized the importance of a digital ITCP-based safety monitoring system.

Recently, the YOLO model has been employed with Unity to present a very effective model for object detection in games [25].

3. Collecting Materials

3.1. Selected Deep Object Detection Algorithms

We found many excellent deep object detection algorithms in the recent literature. Among these algorithms, we selected the most highly-cited algorithms: YOLO [1,10,11], R-CNN [2,12,13], and SSD [3]. Among various versions of YOLO algorithms, we selected YOLOv3 [1], which detects 9000 objects very effectively. For R-CNN algorithms, we selected Faster R-CNN [2], the cutting-edge version of the R-CNNs. Although these algorithms are highly cited, we needed to select a recent algorithm. Therefore, we selected EfficientDet [5]. Therefore, we compared four deep object detection algorithms in our study: YOLOv3, Faster R-CNN, SSD and EfficientDet. The architectures of these algorithms are compared in Figure 1.

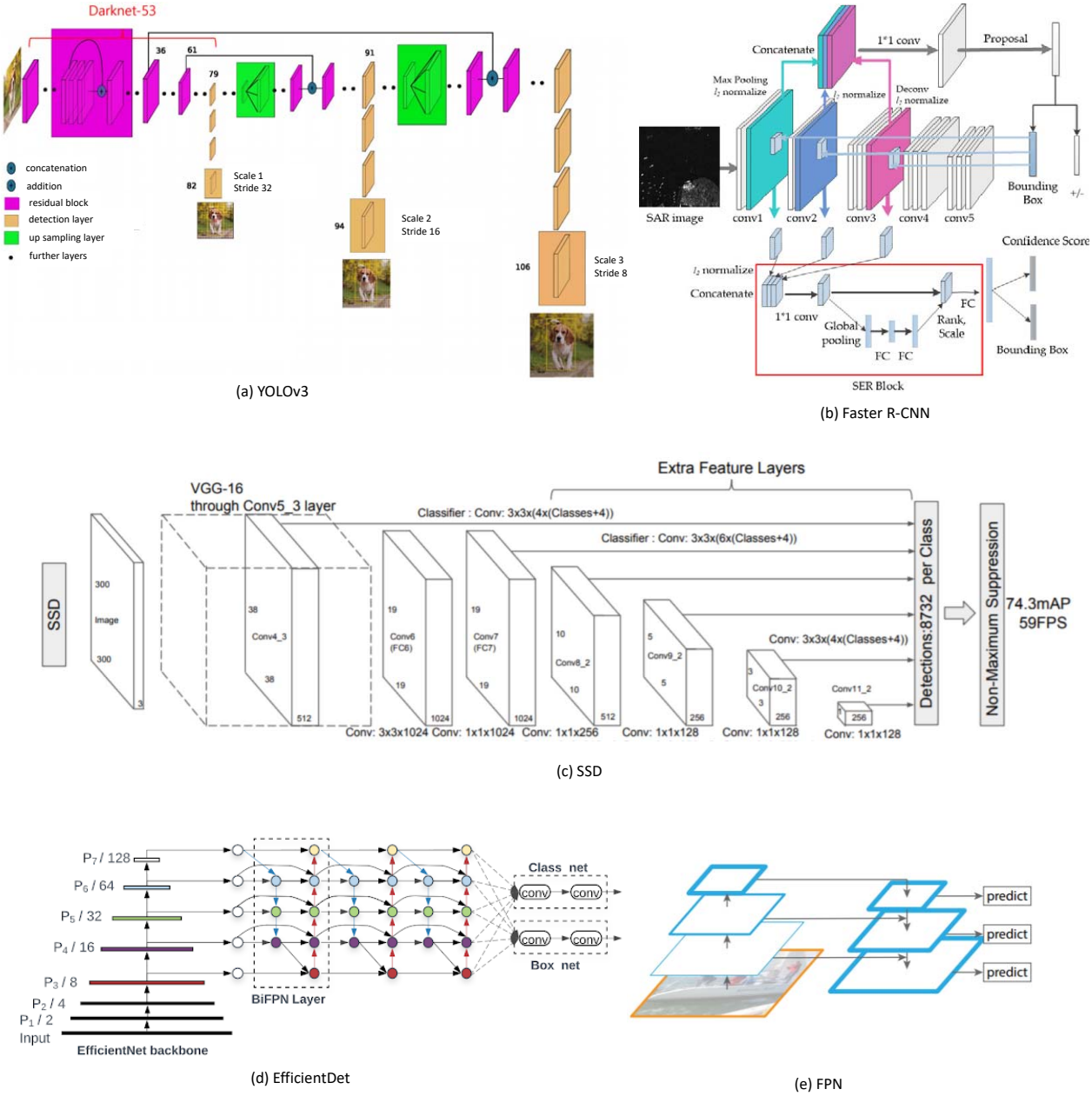


Figure 1. The architectures of the deep object detection algorithms used in this study.

3.2. Selected Games

We had three strategies for selecting games in our study. The first strategy was to select games over various game genres. Therefore, we referred to Wikipedia [26] and sampled game genres including action, adventure, role-playing, simulation, strategy, and sports. The second strategy was to exclude games with objects that existing object detection algorithms cannot recognize. Many role-playing games include fantasy items such as dragons, wyverns, titans, or orcs, which are not recognized by existing algorithms. We also excluded strategy games since they include weapons such as tanks, machine guns, and jet fighters that are not recognized. Our third strategy was to sample both photo-realistically rendered games and cartoon-rendered games. Although most games are rendered photo-realistically, some games employ cartoon-styled rendering because of their uniqueness. Games whose original story is based on cartoons tend to preserve cartoon-styled rendering. Therefore, we sampled cartoon-rendered games to test how the selected algorithms can detect cartoon-styled objects.

We selected games for our study from these genres as evenly as possible. For action and adventure games, we selected *7 Days to Die* [27], *Left 4 Dead 2* [28] and *Gangstar New Orleans* [29]. For simulation, we selected *Sims4* [30], *Animal Crossing* [31], and *Doraemon* [32]. For sports, we selected *Asphalt 8* [33] and *FIFA 20* [34]. Among these games, *Animal Crossing* and *Doraemon* are rendered in a cartoon style. Figure 2 shows illustrations of the selected games.

Figure 2. Eight games we selected for our study.

4. Training and Results

4.1. Training

We retrained the existing object detection algorithms using two datasets: PascalVOC and game scenes. We sampled 800 game scenes: 100 scenes from 8 games we selected. We augmented the sampled game scenes in various schemes: flipping, rotation, controlling hues and controlling tone. By alternating these augmentation schemes, we could build more than 10,000 game scenes for retraining the selected algorithms.

We trained and tested the algorithms on a personal computer with an Intel Pentium i7 CPU and nVidia RTX 2080 GPU. The time required for re-training the algorithms is presented in Table 1.

Table 1. Time required for retraining the algorithms (hrs).

Algorithm	YOLOv3 [1]	Faster R-CNN [2]	SSD [3]	FPN [4]	EfficientDet [5]
Time required for retraining the algorithms	9.5	8.1	9.1	9.4	8.5

4.2. Results

The result images on sampled eight samples comparing pre-trained algorithms and re-trained algorithms are presented in Appendix A. We have presented our results according to the following strategies: recognition performance measured by mAP, localization performance measured by IoU and various statistics. We measured mAP, IoU and various statistic values including average IoU, precision, recall, F1 score and accuracy for the five object recognition algorithms with two datasets.

4.2.1. Measuring and Comparing Recognition Performance Using mAP

In Table 2, we compare mAP values for the five algorithms between the Pascal VOC dataset and the Pascal VOC dataset with game scenes. We show the same comparison on

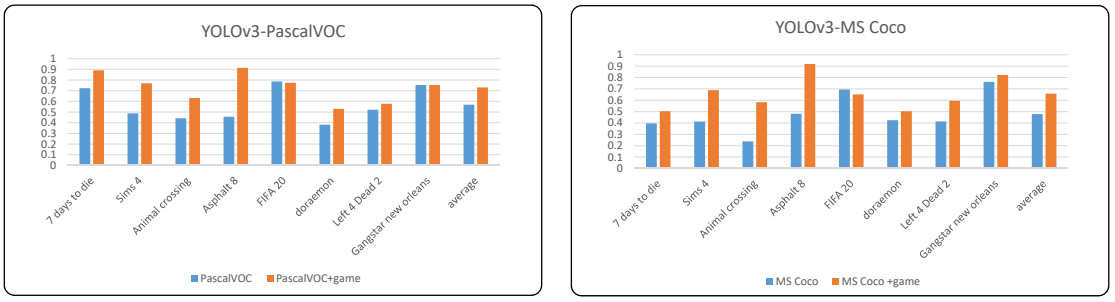
the MS COCO dataset in Table 3. In Figure 3, we illustrate the comparisons presented in Tables 2 and 3.

Table 2. The comparison of mAPs for each game. We compared five object detection algorithms pre-trained by PascalVOC and retrained by PascalVOC with game scenes. Note that PascalVOC is abbreviated as Pascal in the table.

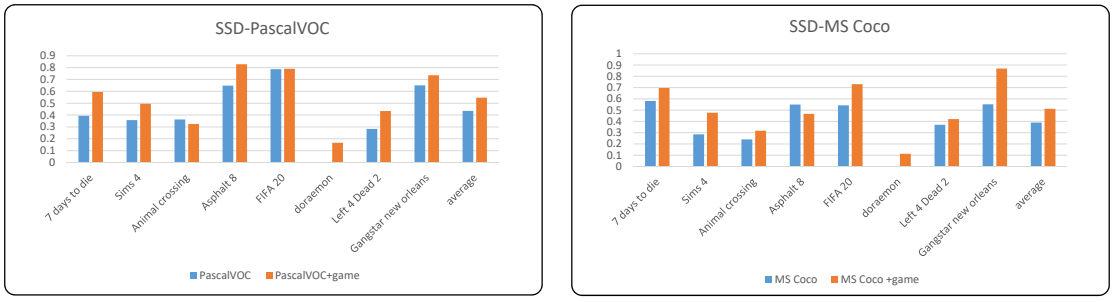
Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**
7 Days to Die	1.0	1.0	0.4167	0.6667	0.5	0.4792	0.5940	0.7376	0.4167	0.75
Sims4	0.7556	0.9182	0.5118	0.7452	0.3885	0.4973	0.75	0.8	0.5092	0.8878
Animal Crossing	0.8389	0.7567	0.65	0.6667	0.4833	0.6167	0.4636	0.7882	0.5524	0.85
Asphalt8	0.375	1.0	0.75	1.0	0.1875	0.725	0.75	1	0.375	0.875
FIFA 20	1.0	1.0	0.8725	0.7857	0.6428	0.7143	0.6694	0.75	1.0	1.0
Doraemon	0.5222	0.8778	0.2056	0.57	0.3167	0.4667	0.1333	0.875	0.1667	0.9444
Left 4 Dead 2	0.9444	0.9724	0.1944	0.67	0.056	0.215	0.5	0.875	0.2222	1.0
Gangstar	1.0	1.0	0.5833	0.9583	0.6667	0.4431	1	1	0.8333	0.9167
Average	0.8045	0.9406	0.5230	0.7578	0.4051	0.4973	0.6076	0.8376	0.5092	0.9029

Table 3. The comparison of mAPs for each game. We compared five object detection algorithms pre-trained by MS COCO and retrained by MS COCO with game scenes. Note that MS COCO is abbreviated as MS in the table.

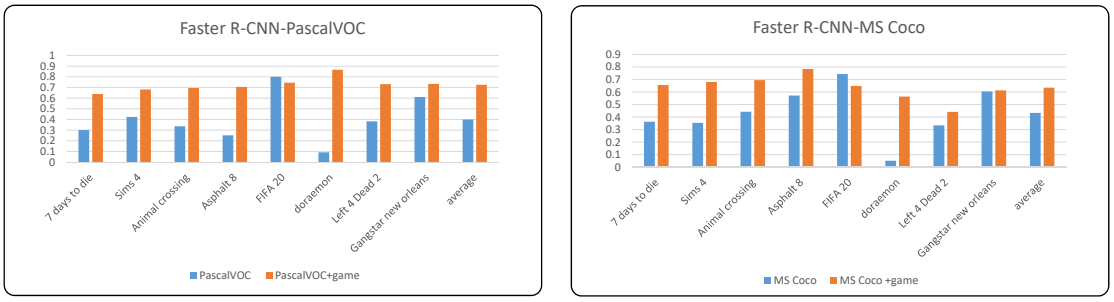
Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**
7 Days to Die	0.88	0.9	0.75	1	0.7457	0.8	0.5	0.8	0.4792	0.9
Sims4	0.3511	0.7984	0.45	0.5961	0.5643	0.725	0.6697	0.8361	0.75	1
Animal Crossing	0.6731	0.8030	0.6694	0.7576	0.4636	0.639	0.4337	0.8394	0.5758	0.875
Asphalt8	0.7781	1	0.75	1	0.6964	0.875	0.6786	0.875	0.7143	1
FIFA 20	0.8712	0.9286	0.748	1	0.8571	0.7857	1	1	0.8571	1
Doraemon	0.3871	0.7202	0	0.5523	0.2762	0.526	0.1333	0.6667	0.4	0.8
Left 4 Dead 2	0.6153	0.9218	0.5833	0.814	0.2857	0.4464	0.875	1	0.875	0.875
Gangstar	0.8914	1	0.4431	0.9583	0.9135	0.9689	0.579	0.875	1	0.8571
Average	0.6809	0.8840	0.54923	0.8348	0.6003	0.7208	0.6087	0.8615	0.7064	0.9134



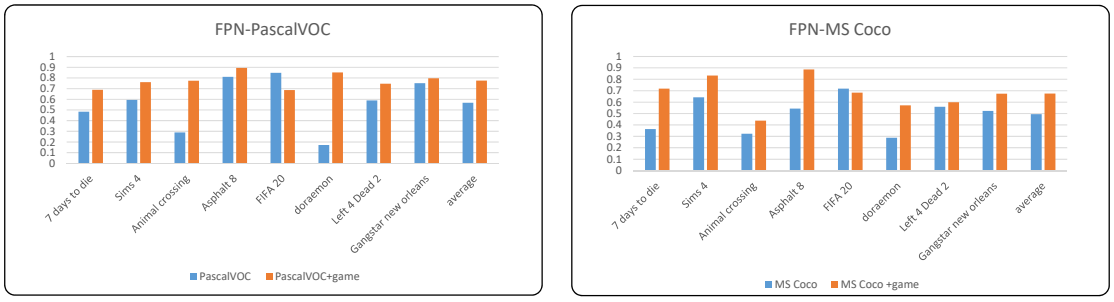
(a) mAP's of YOLOv3 compared by PascalVOC and MS Coco datasets



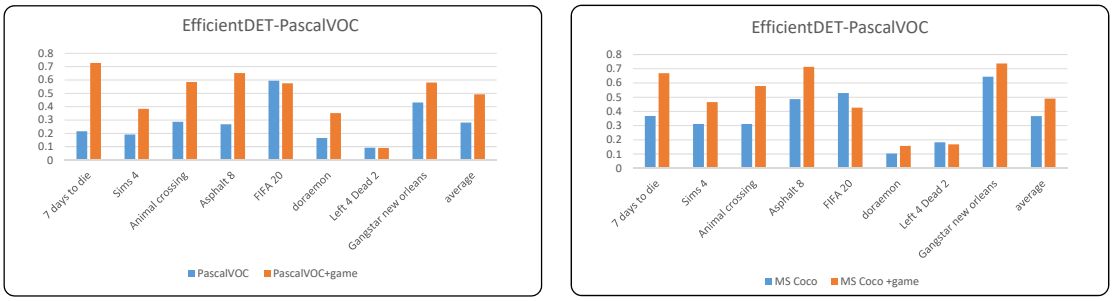
(b) mAP's of SSD compared by PascalVOC and MS Coco datasets



(c) mAP's of Faster R-CNN compared by PascalVOC and MS Coco datasets



(d) mAP's of FPN compared by PascalVOC and MS Coco datasets



(e) mAP's of EfficientDET compared by PascalVOC and MS Coco datasets

Figure 3. mAPs from five object detection algorithms trained by different datasets are compared. In the left column, blue bars denote mAPs from those models trained using PascalVOC only and red bars are for PascalVOC + game scenes. In the right column, blue bars denote mAPs from those models trained using MS COCO only and red bars are for MS COCO + game scenes.

4.2.2. Measuring and Comparing Localization Performance Using IoU

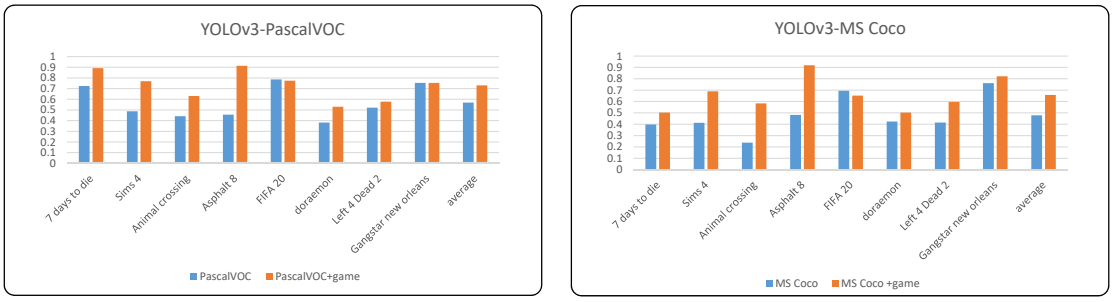
In Table 4, we compare IoU values of the five algorithms between the Pascal VOC dataset and the Pascal VOC dataset with game scenes. We show the same comparison on the MS COCO dataset in Table 5. In Figure 4, we illustrate the comparisons presented in Tables 4 and 5.

Table 4. The comparison of IoUs for each game. We compared five object detection algorithms pre-trained by PascalVOC and retrained by PascalVOC with game scenes. Note that PascalVOC is abbreviated as Pascal in the table.

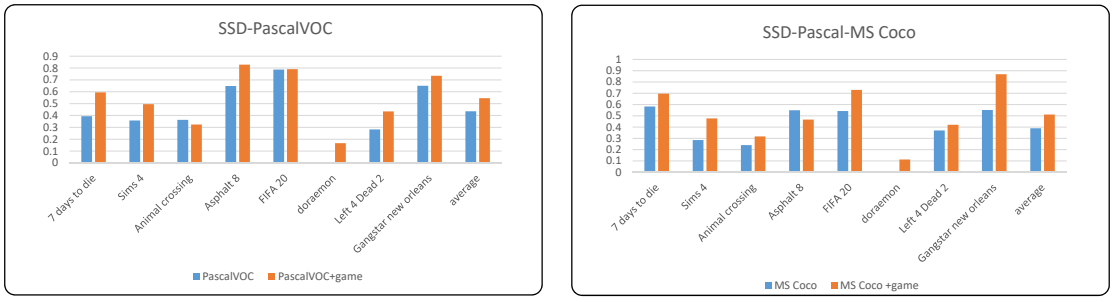
Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**
7 Days to Die	0.7237	0.7889	0.3938	0.6944	0.3010	0.6389	0.4831	0.6891	0.3887	0.5566
Sims4	0.5148	0.6373	0.3576	0.5955	0.4234	0.6805	0.5934	0.7605	0.1911	0.4830
Animal Crossing	0.5001	0.6810	0.3632	0.3545	0.3369	0.6949	0.2891	0.7732	0.3985	0.5974
Asphalt8	0.5560	0.7941	0.6482	0.8289	0.2516	0.7057	0.8099	0.8946	0.3506	0.6062
FIFA 20	0.7857	0.7741	0.7870	0.8500	0.8005	0.7445	0.8472	0.6860	0.5950	0.5752
Doraemon	0.3818	0.6303	0.1664	0.6231	0.0927	0.8656	0.1719	0.8507	0.1644	0.4516
Left 4 Dead 2	0.5214	0.6769	0.2828	0.6345	0.3825	0.7308	0.5894	0.7454	0.0917	0.0905
Gangstar	0.7536	0.8033	0.6503	0.7352	0.6113	0.7334	0.7503	0.7966	0.4308	0.4751
Average	0.5689	0.7299	0.4354	0.5462	0.3999	0.7243	0.2803	0.4935	0.5668	0.7745

Table 5. The comparison of IoUs for each game. We compared five object detection algorithms pre-trained by MS COCO and retrained by MS COCO with game scenes. Note that MS COCO is abbreviated as MS in the table.

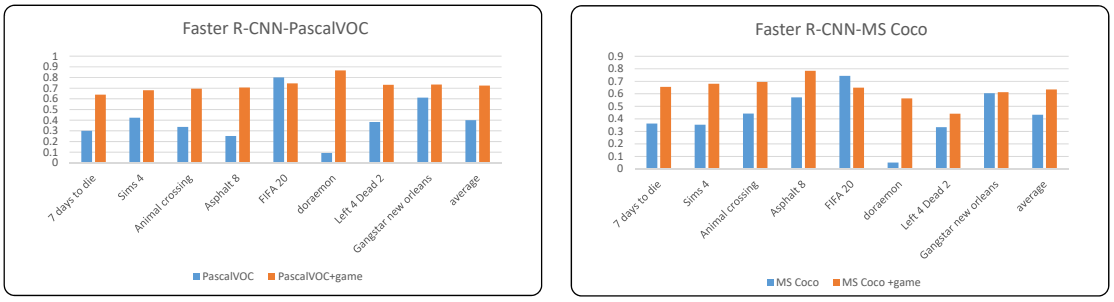
Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**
7 Days to Die	0.3972	0.5024	0.5823	0.6964	0.3624	0.6551	0.3672	0.6691	0.3643	0.7183
Sims4	0.4129	0.6891	0.2852	0.4767	0.3531	0.6805	0.3114	0.4648	0.6419	0.8419
Animal Crossing	0.2382	0.5830	0.2403	0.3174	0.4427	0.6949	0.3109	0.5789	0.3234	0.4371
Asphalt8	0.4817	0.9184	0.5495	0.4665	0.5717	0.7832	0.4858	0.7133	0.5428	0.8873
FIFA 20	0.6942	0.6516	0.5424	0.7304	0.7435	0.6486	0.5299	0.4262	0.7177	0.6830
Doraemon	0.4242	0.5027	0	0.1132	0.0513	0.5632	0.1031	0.1574	0.2873	0.5719
Left 4 Dead 2	0.4144	0.5956	0.3706	0.4209	0.3337	0.4408	0.1824	0.1684	0.5588	0.5985
Gangstar	0.7611	0.8215	0.5515	0.8694	0.6043	0.6125	0.6448	0.7374	0.5224	0.6730
Average	0.4779	0.6580	0.3902	0.5114	0.4328	0.6349	0.3669	0.4894	0.4948	0.6751



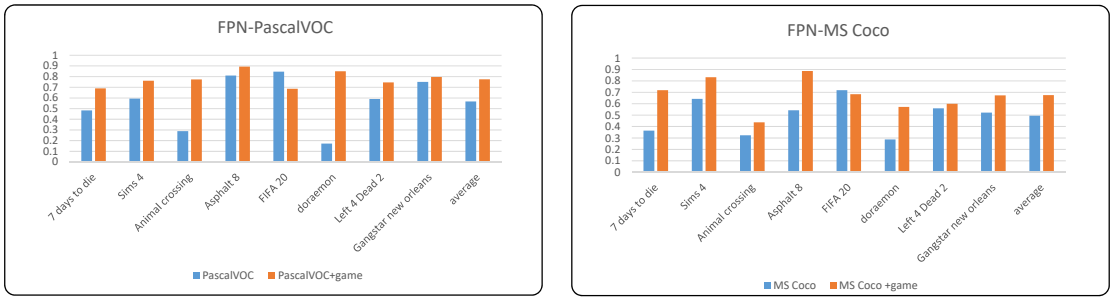
(a) IOU's of YOLOv3 compared by PascalVOC and MS Coco datasets



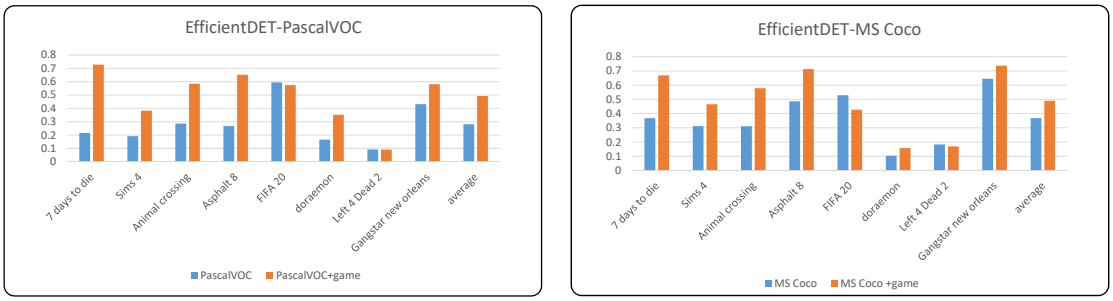
(b) IOU's of SSD compared by PascalVOC and MS Coco datasets



(c) IOU's of Faster R-CNN compared by PascalVOC and MS Coco datasets



(d) IOU's of FPN compared by PascalVOC and MS Coco datasets



(e) IOU's of EfficientDET compared by PascalVOC and MS Coco datasets

Figure 4. IoUs from five object detection algorithms trained by different datasets are compared. In the left column, blue bars denote IoUs from those models trained using PascalVOC only and red bars are for PascalVOC + game scenes. In the right column, blue bars denote IoUs from those models trained using MS COCO only and red bars are for MS COCO + game scenes.

4.2.3. Measuring and Comparing Various Statistics

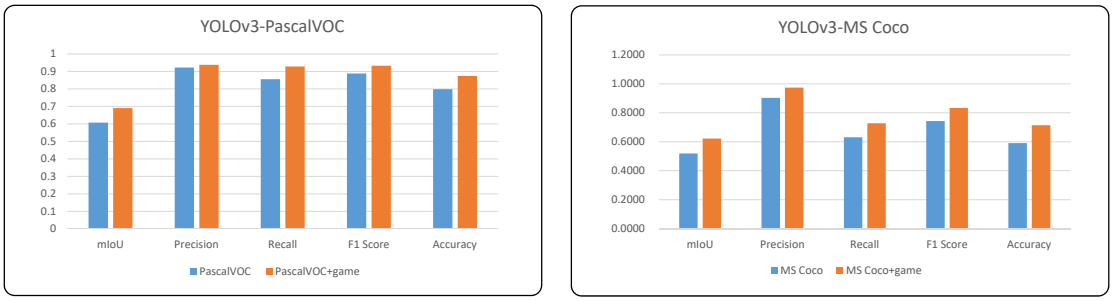
In Tables 6 and 7, we estimate the average IoU, precision, recall, F1 score and accuracy of the five algorithms for the Pascal VOC dataset and the MS COCO dataset. In Figure 5, we illustrate the comparisons presented in Tables 6 and 7.

Table 6. The statistics. We compared the algorithms trained by PascalVOC and retrained by PascalVOC with game scenes for five object detection algorithms. Note that PascalVOC is abbreviated as Pascal in the table.

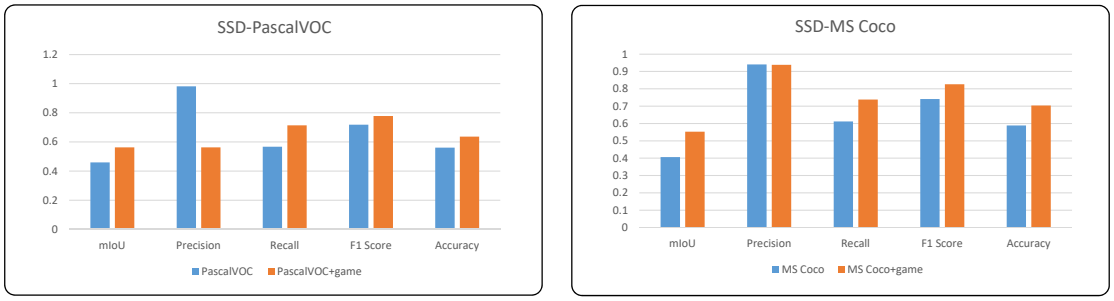
Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**
average IoU	0.5689	0.7299	0.4354	0.5462	0.3999	0.7243	0.2803	0.4934	0.3263	0.4795
precision	0.9222	0.9375	0.9821	0.5627	0.9333	0.9255	0.8571	0.9896	0.6508	0.7763
recall	0.8557	0.9278	0.5670	0.7143	0.5773	0.8969	0.6990	0.9223	0.4227	0.6082
F1 score	0.8877	0.9326	0.7190	0.7778	0.7134	0.9110	0.7701	0.9548	0.5125	0.6821
accuracy	0.7981	0.8738	0.5612	0.6364	0.5545	0.8365	0.6261	0.9135	0.3445	0.5175

Table 7. The statistics. We compared the algorithms trained by MS COCO and retrained by MS COCO with game scenes for five object detection algorithms. Note that MS COCO is abbreviated as MS in the table.

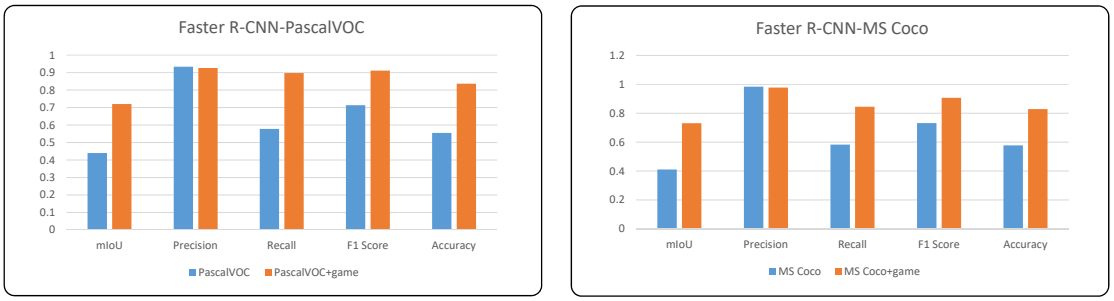
Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**
average IoU	0.4779	0.6580	0.3902	0.5114	0.4328	0.6349	0.3669	0.4894	0.4948	0.6751
precision	0.9028	0.9740	0.9403	0.9383	0.9836	0.9775	0.9231	0.9870	0.8364	0.8772
recall	0.6311	0.7282	0.6117	0.7379	0.5825	0.8447	0.6990	0.7379	0.4466	0.4854
F1 score	0.7429	0.8333	0.7412	0.8261	0.7317	0.9062	0.7956	0.8444	0.5823	0.6250
accuracy	0.5909	0.7143	0.5888	0.7037	0.5769	0.8286	0.6606	0.7308	0.4107	0.4545



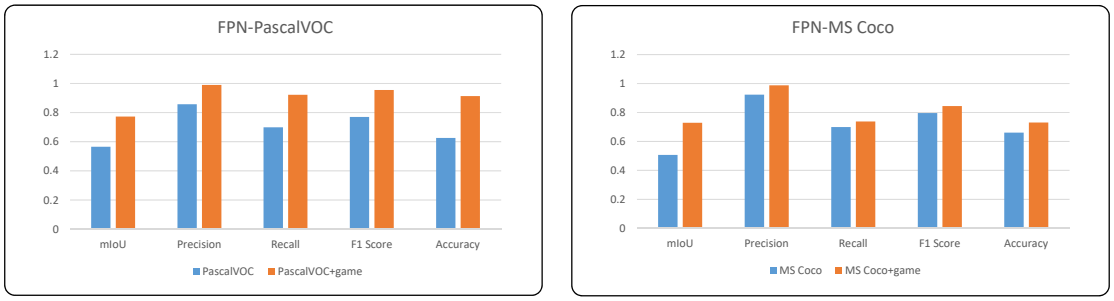
(a) statistics of YOLOv3 compared by PascalVOC and MS Coco datasets



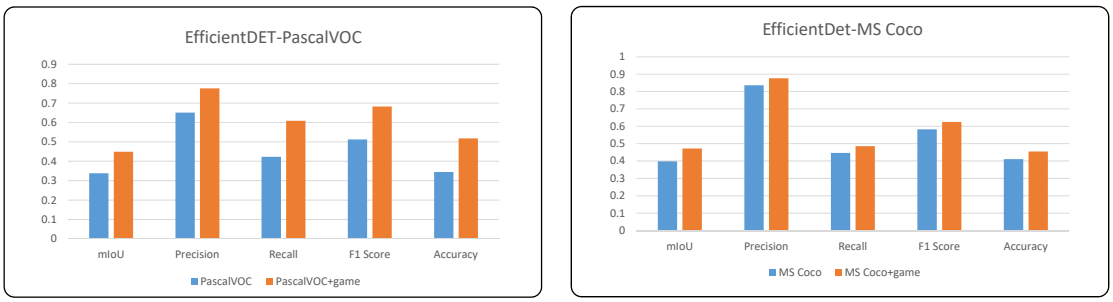
(b) statistics of SSD compared by PascalVOC and MS Coco datasets



(c) statistics of Faster R-CNN compared by PascalVOC and MS Coco datasets



(d) statistics of FPN compared by PascalVOC and MS Coco datasets



(e) statistics of EfficientDET compared by PascalVOC and MS Coco datasets

Figure 5. Mean IoU, precision, recall, F1 score and accuracy are compared between two different datasets. In the left column, blue bars denote the values from those models trained using PascalVOC only and red bars are for PascalVOC + game scenes. In the right column, blue bars denote the values from those models trained using MS COCO only and red bars are for MS COCO + game scenes.

5. Analysis

To prove our claim that the object detection algorithms retrained with game scenes show better performance than the object detection algorithms trained only with existing datasets such as Pascal VOC and MS COCO, we asked the following research questions (*RQ*).

*RQ*1. Does our strategy to retrain existing object detection algorithms with game scenes improve mAP?

*RQ*2. Does our strategy to retrain existing object detection algorithms with game scenes improve IoU?

5.1. Analysis of mAP Improvement

To answer *RQ*1, we compared and analyzed mAP values suggested in Tables 2 and 3, which compare the mAP values of the object detection algorithms trained only with the existing datasets and retrained with game scenes. An overall observation reveals that the retrained object detection algorithms show better mAP than the pre-trained algorithms for 61 of all 80 cases. For further analysis, we performed a *t*-test and measured the effect size using Cohen's *d* value.

5.1.1. *t*-Test

Table 8 compares the p values for the five algorithms trained by PascalVOC and retrained by PascalVOC + game scenes. From the p values, we found that the results from three of the five algorithms are distinguished for $p < 0.05$. The results from EffficientDet are distinguished even for $p < 0.01$.

Table 9 compares the p values for the five algorithms trained by MS COCO and retrained by MS COCO + game scenes. From the p values, we found that the results from four of the five algorithms are distinguished for $p < 0.05$.

From these results, we show that seven cases from all ten cases exhibit significantly distinguishable results for $p < 0.05$.

5.1.2. Cohen's *d*

We also measured the effect size using Cohen's *d* value for the mAP values and present the results in Tables 10 and 11.

Since four Cohen's *d* values in Table 10 are greater than 0.8, we can conclude that the effect size of retraining the algorithms using game scenes is great for four algorithms.

We also suggest the Cohen's *d* values measured from the MS COCO dataset in Table 11, where four Cohen's *d* values are greater than 0.8. We can also conclude that the effect size of retraining the algorithms using game scenes is great for four algorithms.

Table 8. p values for the mAPs from five object detection algorithms. We compared the algorithms trained by PascalVOC and retrained by PascalVOC with game scenes. Note that PascalVOC is abbreviated as Pascal in the table.

Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**
average mAP	0.8045	0.9182	0.5118	0.7452	0.3885	0.4973	0.6076	0.8376	0.5092	0.8878
std. dev.	0.2396	0.1057	0.2457	0.1562	0.2196	0.2114	0.2552	0.1091	0.2864	0.0959
p	0.2476		0.0397		0.3301		0.0437		0.0063	
$p < 0.05$	Not		Distinguished		Not		Distinguished		Distinguished	
$p < 0.01$	Not		Not		Not		Not		Distinguished	

Table 9. p values for the mAPs from five object detection algorithms. We compared the algorithms trained by MS COCO and retrained by MS COCO with game scenes. Note that MS COCO is abbreviated as MS in the table.

Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**
average mAP	0.6809	0.8840	0.5492	0.8348	0.6003	0.7208	0.7064	0.9134	0.6087	0.8615
std. dev.	0.2168	0.1009	0.2559	0.1853	0.2445	0.1756	0.2076	0.0771	0.2678	0.1079
p	0.0372		0.0228		0.2768		0.0352		0.0268	
$p < 0.05$	Distinguished		Distinguished		Not		Distinguished		Distinguished	
$p < 0.01$	Not		Not		Not		Not		Not	

Table 10. Cohen's d values for mAPs from five object detection algorithms. We compared the algorithms trained by PascalVOC and retrained by PascalVOC with game scenes. Note that PascalVOC is abbreviated as Pascal in the table.

Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**
average mAP	0.8045	0.9182	0.5118	0.7452	0.3885	0.4973	0.6076	0.8376	0.5092	0.8878
std. dev.	0.1883		0.2326		0.2743		0.2157		0.2237	
Cohen's d	0.8681		1.6031		2.5073		1.0278		0.7136	
Effect size	Large		Large		Large		Large		>medium	

Table 11. Cohen's d values for mAPs from five object detection algorithms. We compared the algorithms trained by MS COCO and retrained by MS COCO with game scenes. Note that MS COCO is abbreviated as MS in the table.

Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**
average mAP	0.6809	0.8840	0.5492	0.8348	0.6003	0.7208	0.7064	0.9134	0.6087	0.8615
std. dev.	0.1941		0.2614		0.2148		0.1853		0.2365	
Cohen's d	1.0461		1.0924		1.0689		1.1171		0.5606	
Effect size	Large		Large		Large		Large		>medium	

5.2. Analysis on the Improvement of IoU

To answer *RQ*2, we compared and analyzed IoU values suggested in Tables 4 and 5 that compare the IoU values of the object detection algorithms trained only with existing datasets and retrained with game scenes. From these values, we found that the retrained object detection algorithms show better IoU for 68 of all 80 cases. For further analysis, we performed a t-test and measured the effect size using Cohen's d value.

5.2.1. t-Test

Table 12 compares the p values for the five algorithms trained by PascalVOC and PascalVOC + game scenes. From the p-values, we found that the results from all the five

algorithms are distinguished for $p < 0.05$. Therefore, our strategy to retrain the algorithms with game scenes shows a significant improvement for localization.

Table 13 compares the p values for the five algorithms trained by MS COCO and MS COCO + game scenes. From the p-values, we found that the results from three algorithms are distinguished for $p < 0.05$.

From these results, we have demonstrated that eight cases from all ten cases show a significant distinguishable results for $p < 0.05$.

Table 12. p values for the IoUs from four object detection algorithms. We compared the algorithms trained by PascalVOC and retrained by PascalVOC with game scenes. Note that PascalVOC is abbreviated as Pascal in the table.

Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**	**Pascal**	**Pascal + Game**
average IoU	0.5689	0.7299	0.4356	0.5462	0.3999	0.7243	0.5668	0.7745	0.2803	0.4935
std. dev.	0.1594	0.1398	0.2509	0.2348	0.2191	0.0665	0.2428	0.0726	0.1616	0.2064
p	0.0497		0.0377		0.0039		0.0490		0.0373	
$p < 0.05$	Distinguished		Distinguished		Distinguished		Distinguished		Distinguished	
$p < 0.01$	Not		Not		Distinguished		Not		Not	

Table 13. p values for the IoUs from four object detection algorithms. We compared the algorithms trained by MS COCO and retrained by MS COCO with game scenes. Note that MS COCO is abbreviated as MS in the table.

Algorithm	**YOLOv3**		**SSD**		**Faster R-CNN**		**FPN**		**EfficientDet**	
Dataset	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**	**MS**	**MS + Game**
average IoU	0.4779	0.6580	0.3902	0.5114	0.4329	0.6439	0.3669	0.4894	0.4948	0.6751
std. dev.	0.1699	0.1480	0.2062	0.2443	0.2111	0.1011	0.1804	0.2298	0.1549	0.1438
p	0.0403		0.3021		0.0348		0.2554		0.0301	
$p < 0.05$	Distinguished		Not		Distinguished		Not		Distinguished	
$p < 0.01$	Not		Not		Not		Not		Not	

5.2.2. Cohen's d

We also measured the effect size using Cohen's d value for the IoU values and present the results in Tables 14 and 15.

Since four Cohen's d values in Table 10 are greater than 0.8, we can conclude that the effect size of retraining the algorithms using game scenes is great for four algorithms.

We also suggest the Cohen's d values measured from the MS COCO dataset in Table 11, where three Cohen's d values are greater than 0.8. We can also conclude that the effect size of retraining the algorithms using game scenes is great for three algorithms.

Table 14. Cohen's *d* values for IoUs from four object detection algorithms. We compared the algorithms trained by PascalVOC and retrained by PascalVOC with game scenes. Note that PascalVOC is abbreviated as Pascal in the table.

Algorithm	YOLOv3		SSD		Faster R-CNN		FPN		EfficientDet	
Dataset	Pascal	Pascal + Game	Pascal	Pascal + Game	Pascal	Pascal + Game	Pascal	Pascal + Game	Pascal	Pascal + Game
average IoU	0.5689	0.7299	0.4356	0.5462	0.3999	0.7243	0.5668	0.7745	0.2803	0.4935
std. dev.	0.1670		0.2416		0.2291		0.2037		0.2102	
Cohen's d	0.9641		0.4586		1.4153		1.020		0.0142	
Effect size	Large		>small		Large		Large		Large	

Table 15. Cohen's *d* values for IoUs from four object detection algorithms. We compared the algorithms trained by MS COCO and retrained by MS COCO with game scenes. Note that MS COCO is abbreviated as MS in the table.

Algorithm	YOLOv3		SSD		Faster R-CNN		FPN		EfficientDet	
Dataset	MS	MS + Game	MS	MS + Game	MS	MS + Game	MS	MS + Game	MS	MS + Game
average IoU	0.4779	0.6580	0.3902	0.5114	0.4329	0.6439	0.3669	0.4894	0.4948	0.6751
std. dev.	0.1798		0.2272		0.1909		0.1718		0.2094	
Cohen's d	1.0011		0.5332		1.0581		1.0494		0.5849	
Effect size	Large		>medium		Large		>medium		Large	

In summary, mAP is improved for 61 of 80 cases and IoU for 68 of 80 cases. When we performed a *t*-test on $p < 0.05$, 7 of 10 cases showed a significantly unique improvement for mAP and 8 of 10 cases for IoU. When we measured the effect size, 8 of 10 cases showed a large effect size for mAP and 7 of 10 for IoU. Therefore, we can answer the research questions as the object detection algorithms retrained with game scenes show an improved mAP and IoU compared with the algorithms trained only with public datasets including PascalVOC and MS COCO.

5.3. Training with Augmented Dataset

An interesting approach for improving the performance of object detection algorithms on game scenes is to employ augmented images from datasets such as Pascal VOC or MS COCO. In several studies, intentionally transformed images are generated and employed to train pedestrian detection [35,36]. In our approach, stylization schemes are employed to render images in some game scene style. The stylization schemes we employ include flow-based image abstraction with coherent lines [37], color abstraction using bilateral filters [38] and deep cartoon-styled rendering [39].

In our approach, we augmented 3000 images by applying three stylization schemes [37–39] and retrained object detection algorithms. Some of the augmented images are suggested in Figure 6. In Table 16, we present a comparison between mAP values from pre-trained algorithms and mAP values from retraining with the augmented images. We tested this approach for the Pascal VOC dataset.

Among the eight games we used for the experiment, the scenes from *Doraemon* show similar styles to the augmented images. It is interesting to note that this approach shows somewhat improved results on the scenes from *Doraemon*. For other game scenes, we cannot recognize the improvement in the results. Figure 7 illustrates the comparison of three approaches: (i) trained with Pascal VOC, (ii) retrained with augmentation and (iii) retrained with game scenes.

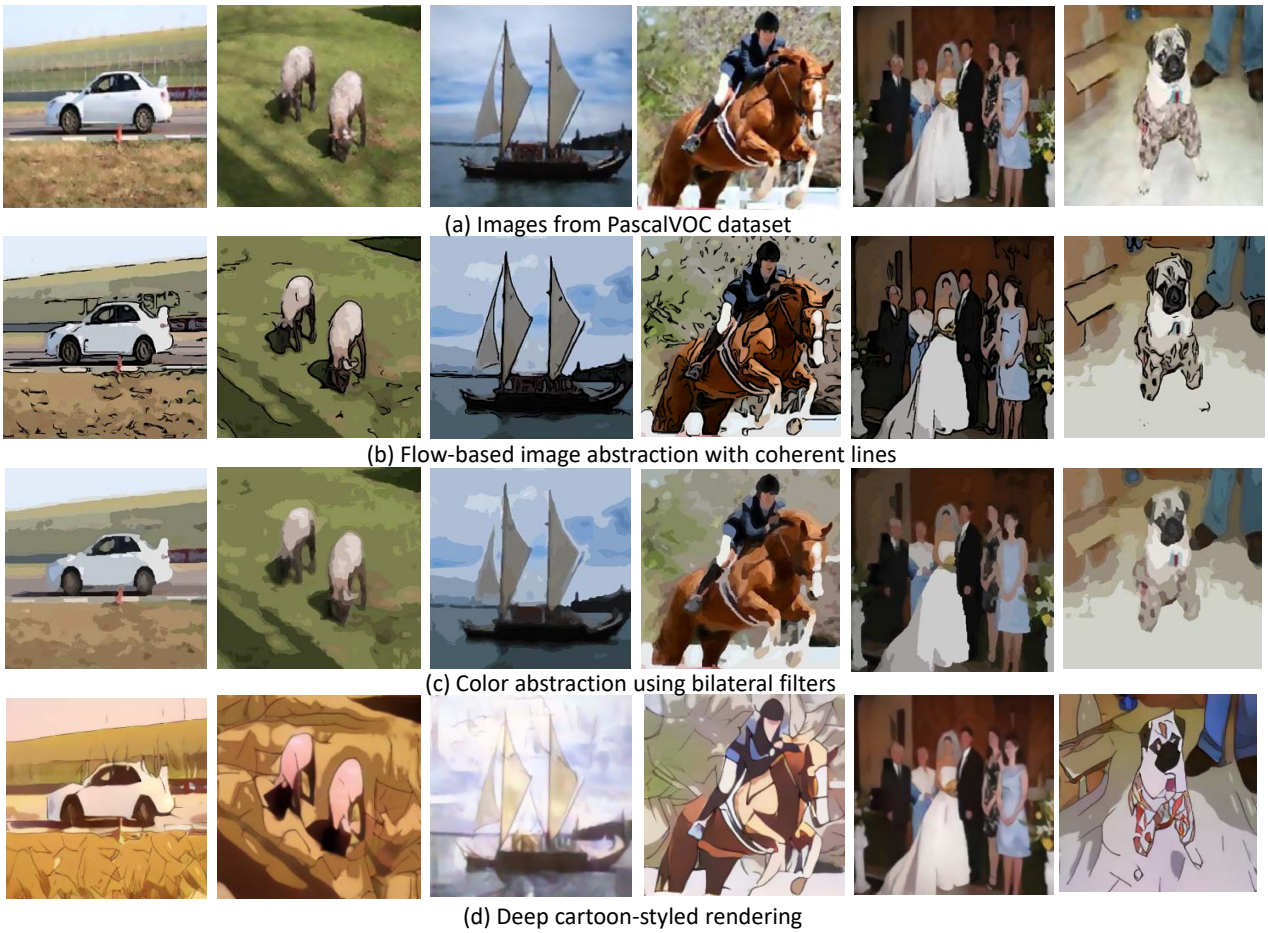


Figure 6. The augmented images from Pascal VOC: (**a**) is the sampled images from Pascal VOC dastaset, (**b**) is produced by flow-based image abstraction with coherent lines [37], (**c**) is produced by color abstraction using a bilateral filter [38] and (**d**) is produced by deep cartoon-styled rendering [39].

Table 16. The comparison of mAPs of the object detection algorithms between the VOC dataset trained with Pascal and retrained using augmented images.

Algorithm	YOLOv3		SSD		Faster R-CNN		FPN		EfficientDet	
Dataset	Pascal	Pascal + Augment	Pascal	Pascal + Augment	Pascal	Pascal + Augment	Pascal	Pascal + Augment	Pascal	Pascal + Augment
7 Days to Die	1.0	1.0	0.4167	0.3705	0.5	0.5413	0.5940	0.5415	0.4167	0.5035
Sims4	0.7556	0.8125	0.5118	0.4915	0.3885	0.4124	0.75	0.8125	0.5092	0.4501
Animal Crossing	0.8389	0.8735	0.65	0.6413	0.4833	0.3314	0.4636	0.5315	0.5524	0.5035
Asphalt8	0.375	0.4214	0.75	0.8215	0.1875	0.3641	0.75	0.8125	0.375	0.5102
FIFA 20	1.0	0.8921	0.8725	0.8613	0.6428	0.6784	0.6694	0.5603	1.0	1.0
Doraemon	0.5222	0.6712	0.2056	0.4510	0.3167	0.5124	0.1333	0.3125	0.1667	0.4315
Left 4 Dead 2	0.9444	0.9214	0.1944	0.315	0.056	0.2105	0.5	0.6124	0.2222	0.3415
Gangstar	1.0	0.9416	0.5833	0.6135	0.6667	0.4641	1	0.9315	0.8333	0.8145
Average	0.8045	0.8168	0.5230	0.5707	0.4051	0.4288	0.6076	0.6394	0.5092	0.5693

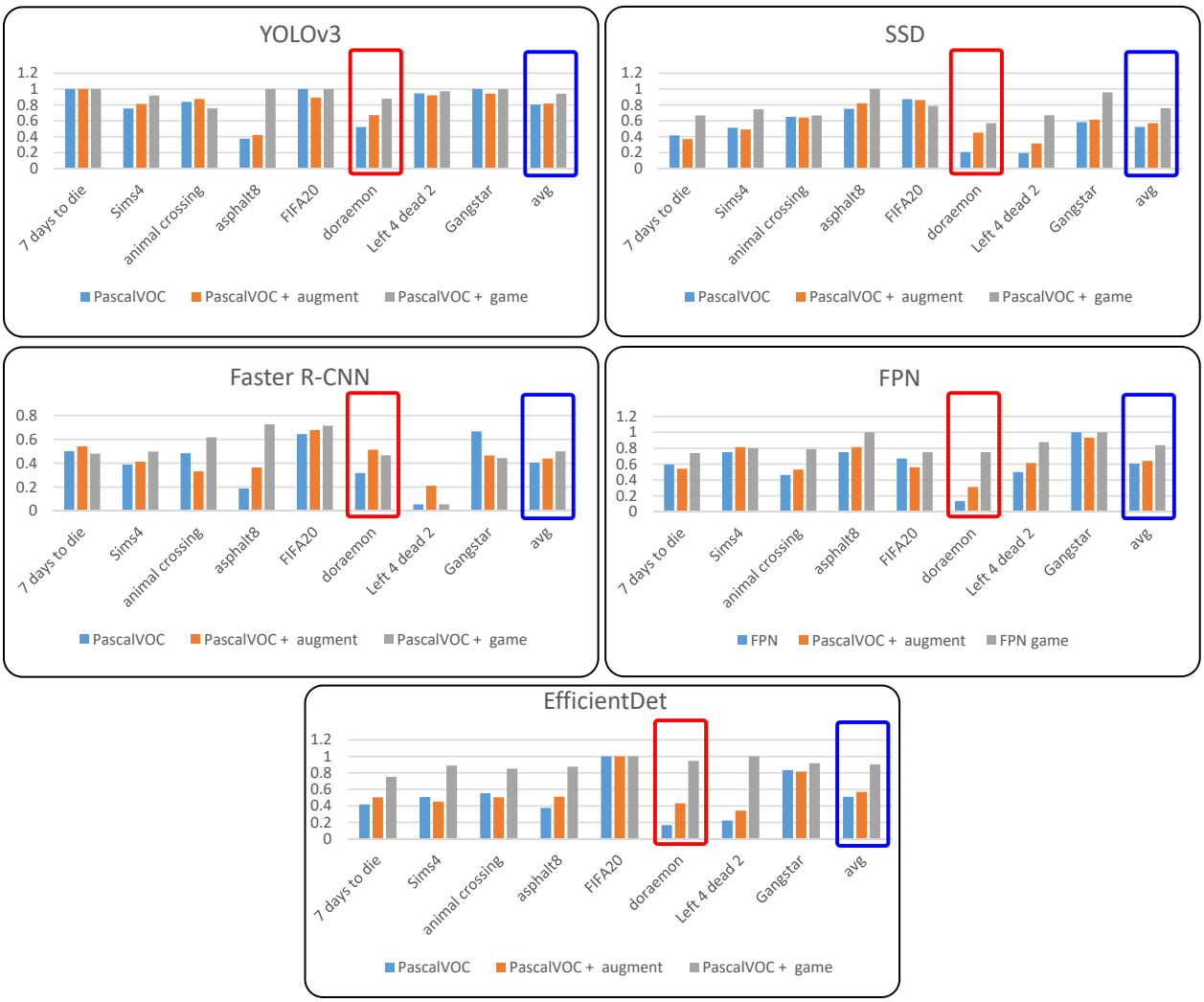


Figure 7. Comparison of three approaches: training only with Pascal VOC, retrained with augmented images and retrained with game scenes. The red rectangle shows comparison of mAPs on scenes from *Doraemon*, which shows greatest improvement. The blue rectangle shows comparison of average mAPs.

6. Conclusions and Future Work

This study proved that the object detection algorithms retrained using game scenes show an improved performance compared with the algorithms trained only with the public datasets. Pascal VOC and MS COCO, two of the most frequently used datasets, were employed for our study. We tested our approach for five widely used object detection algorithms, YOLOv3, SSD, Faster R-CNN, FPN and EfficientDet, and for eight games from various genres. We estimated mAP between the pre-trained and retrained algorithms to show that object recognition accuracy is improved. We also estimated IoU to show that the accuracy of localizing objects is improved. We also tested data augmentation schemes that can be applied for our purpose, which shows very limited results according to the style of game scenes.

We have two further research directions. One direction is to establish a dataset about game scenes to improve the performance of existing object detection algorithms on game scenes. We aim to include various non-existent characters such as dragons, elves or orcs. Another direction is to modify the structure of the object detection algorithms to optimize them on game scenes.

Author Contributions: Conceptualization, M.J. and K.M.; methodology, H.Y.; software, M.J.; validation, H.Y. and K.M.; formal analysis, K.M.; investigation, M.J.; resources, M.J.; data curation, M.J.; writing—original draft preparation, H.Y.; writing—review and editing, K.M.; visualization, K.M.; supervision, K.M.; project administration, K.M.; funding acquisition, K.M. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by Sangmyung Univ. Research Fund 2019.

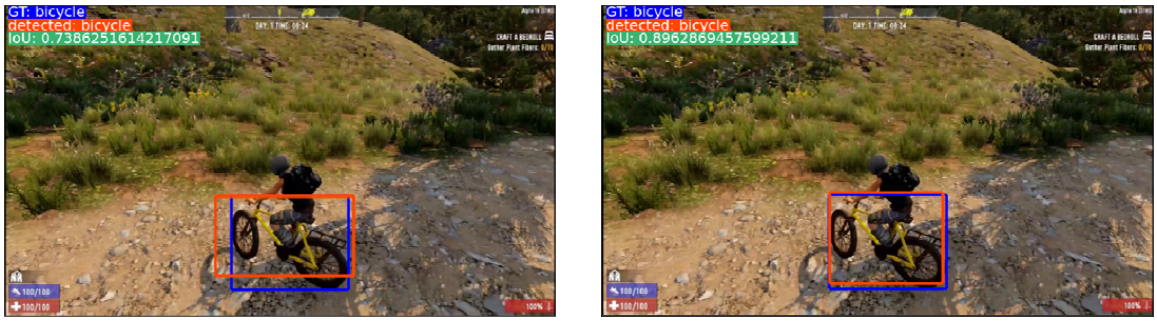
Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In Appendix A, we present eight figures (Figures A1–A8) that sample the results for eight games by five important object detection algorithms: YOLOv3, SSD, Faster R-CNN, FPN and EfficientDet.

(a) The results of YOLOv3

(b) The results of SSD



(c) The results of Faster R-CNN

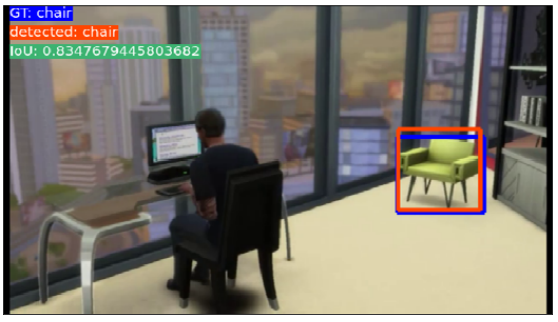
Figure A1. *Cont.*

(d) The results of EfficientDet

(e) The results of FPN

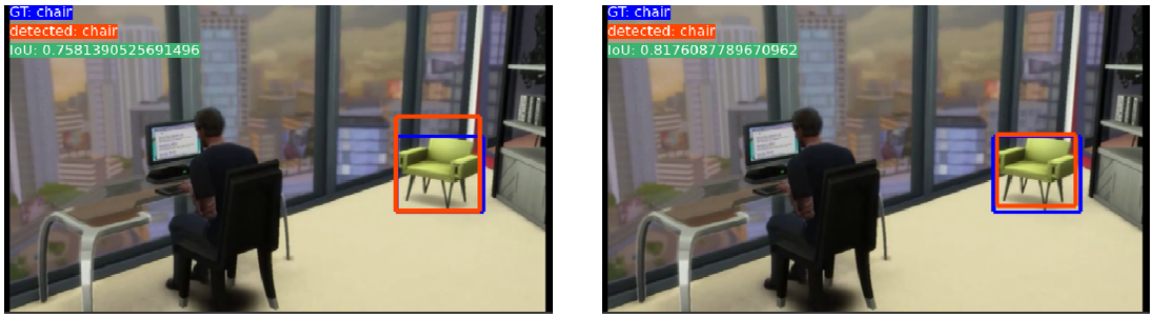
Figure A1. Comparison of the bounding box detection on game *7 Days to Die*. The left column is the result from the models trained by PASCAL VOC and the right column is the result from the models trained by PASCAL VOC and game scenes.

(a) The results of YOLOv3

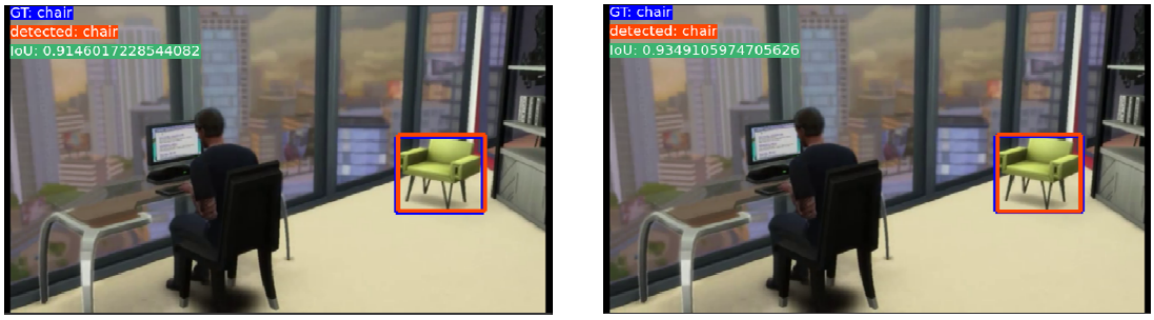


(b) The results of SSD

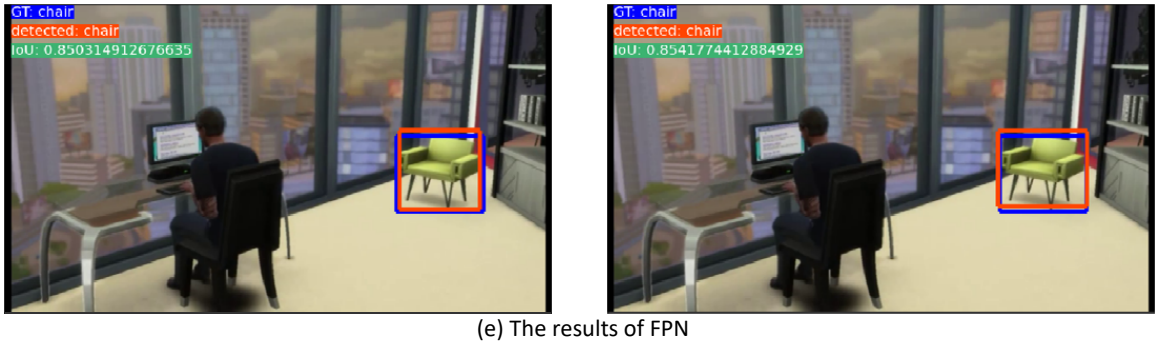
Figure A2. *Cont*.



(c) The results of Faster R-CNN

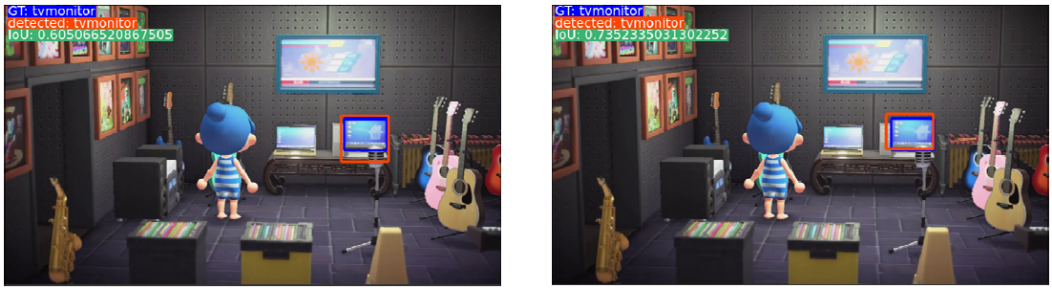


(d) The results of EfficientDet

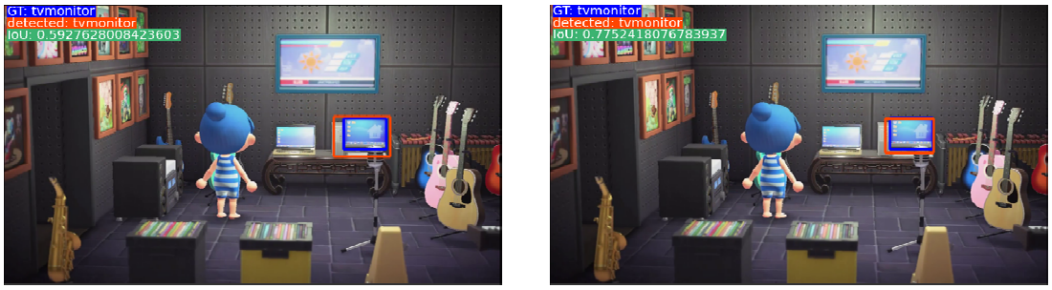


(e) The results of FPN

Figure A2. Comparison of the bounding box detection on game *Sims*. The left column is the result from the models trained by PASCAL VOC and the right column is the result from the models trained by PASCAL VOC and game scenes.



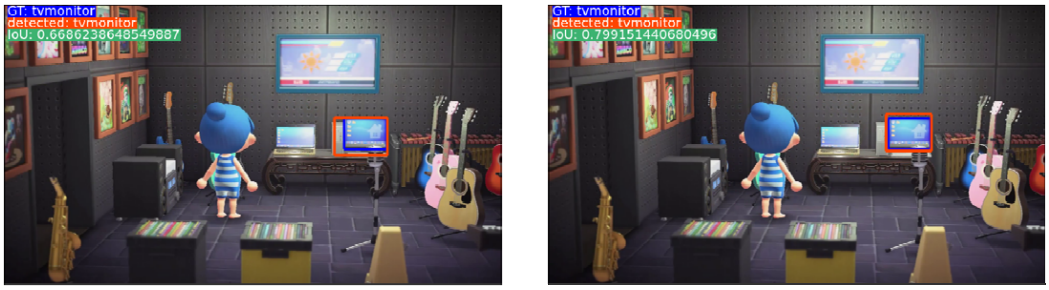
(a) The results of YOLOv3



(b) The results of SSD

(c) The results of Faster R-CNN

(d) The results of EfficientDet



(e) The results of FPN

Figure A3. Comparison of the bounding box detection on game *Animal Crossing*. The left column is the result from the models trained by PASCAL VOC and the right column is the result from the models trained by PASCAL VOC and game scenes.

Figure A4. Comparison of the bounding box detection on game *Asphalt 8*. The left column is the result from the models trained by PASCAL VOC and the right column is the result from the models trained by PASCAL VOC and game scenes.

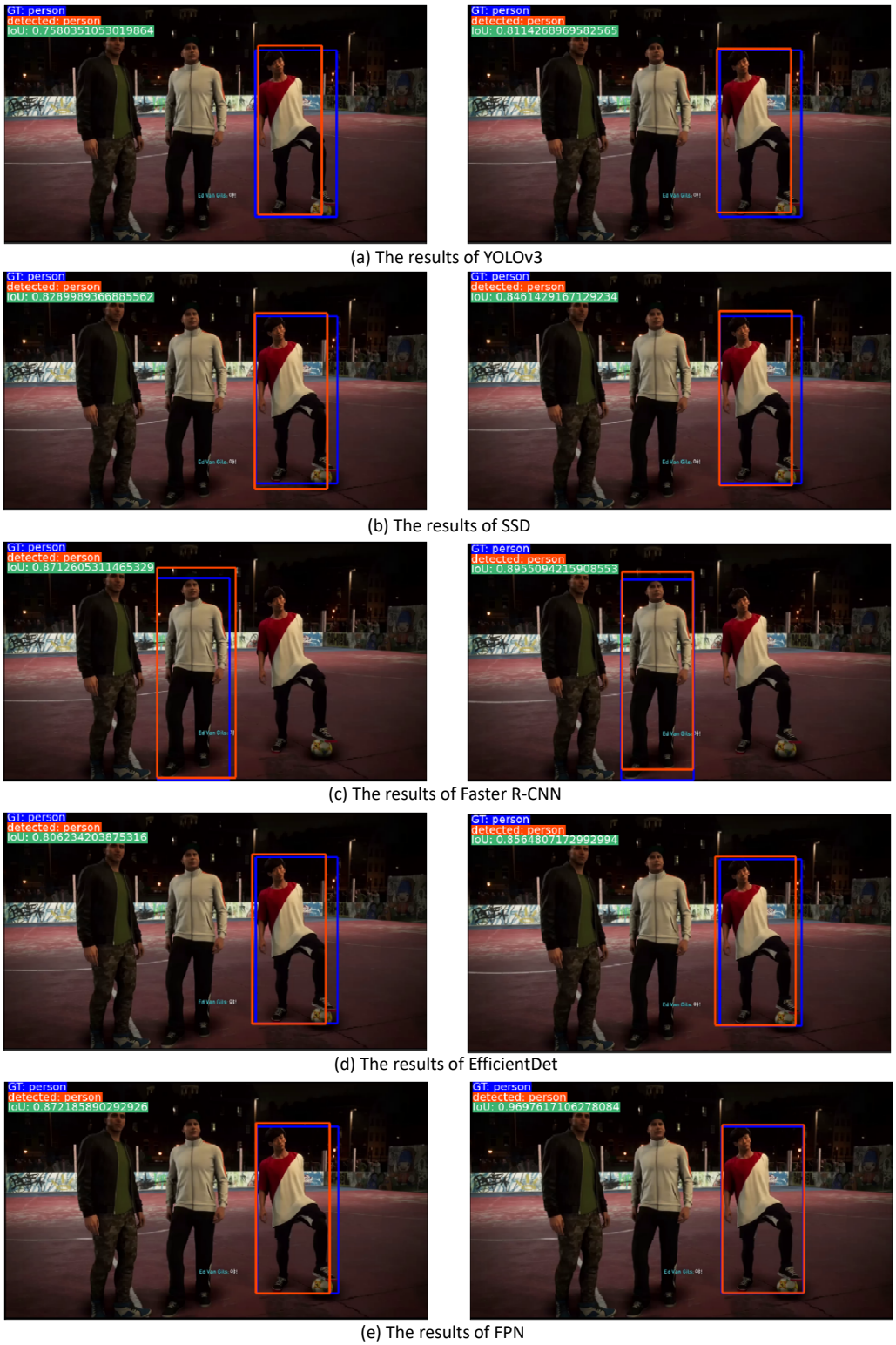


Figure A5. Comparison of the bounding box detection on game *FIFA*20. The left column is the result from the models trained by PASCAL VOC and the right column is the result from the models trained by PASCAL VOC and game scenes.

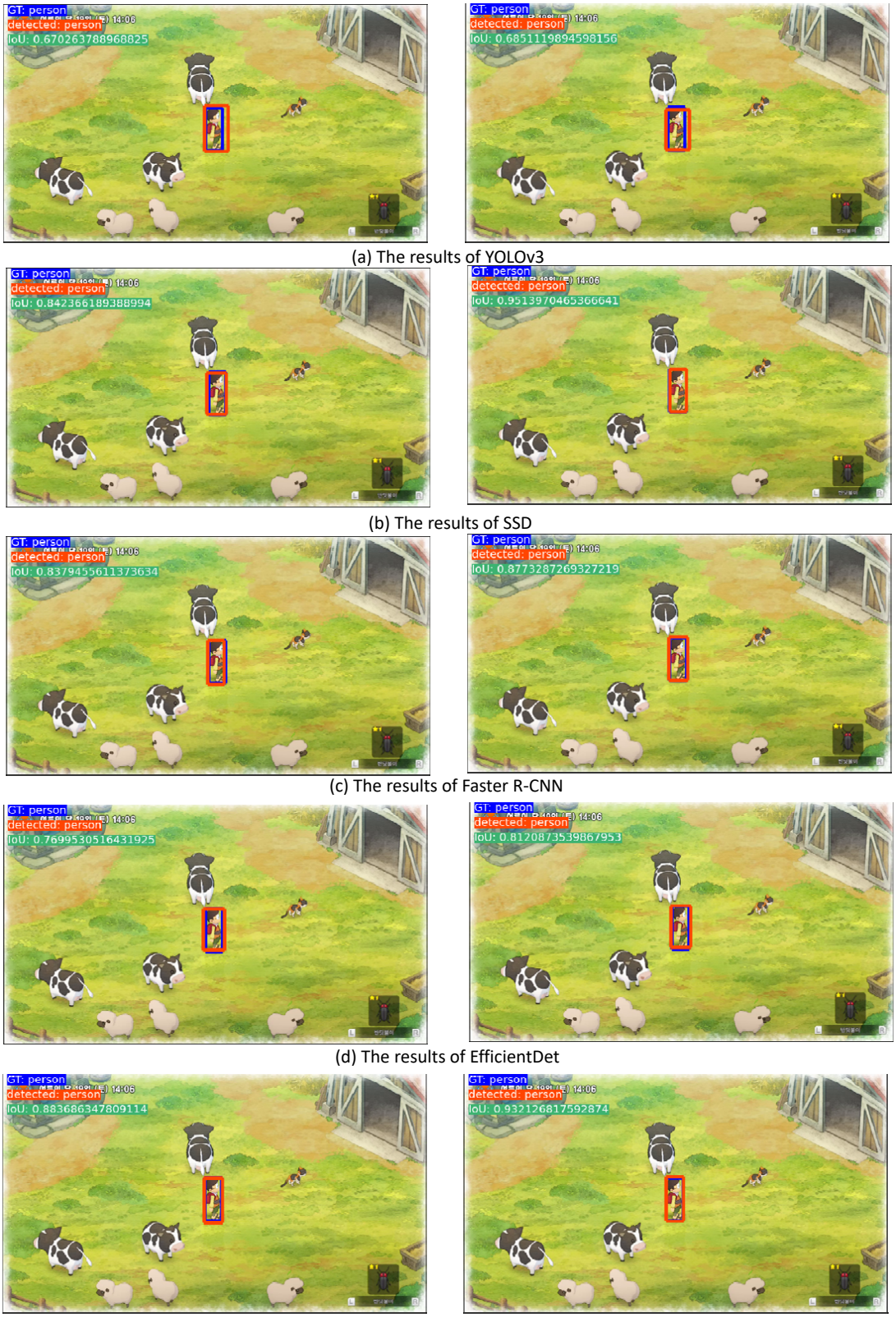


Figure A6. Comparison of the bounding box detection on game *Doraemon*. The left column is the result from the models trained by PASCAL VOC and the right column is the result from the models trained by PASCAL VOC and game scenes.

(a) The results of YOLOv3

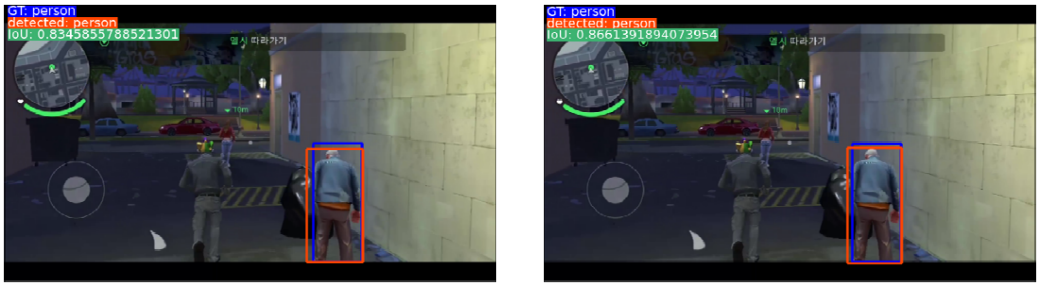
(b) The results of SSD

(c) The results of Faster R-CNN

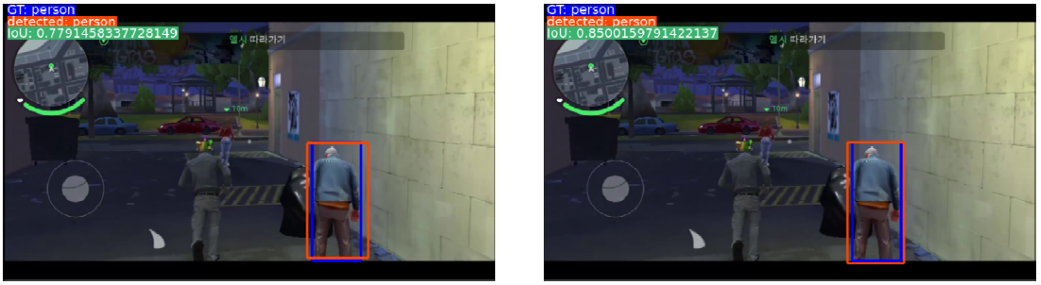
(d) The results of EfficientDet

(e) The results of FPN

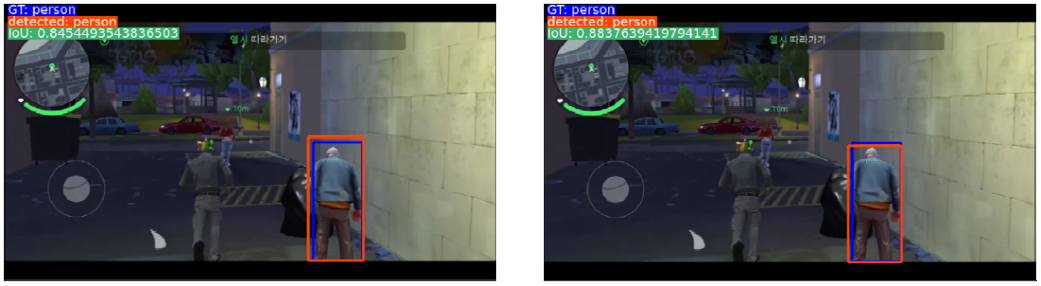
Figure A7. Comparison of the bounding box detection on game *Left4Dead2*. The left column is the result from the models trained by PASCAL VOC and the right column is the result from the models trained by PASCAL VOC and game scenes.



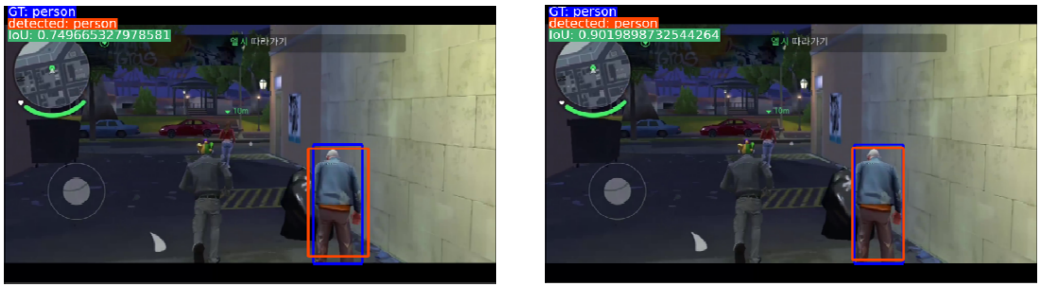
(a) The results of YOLOv3



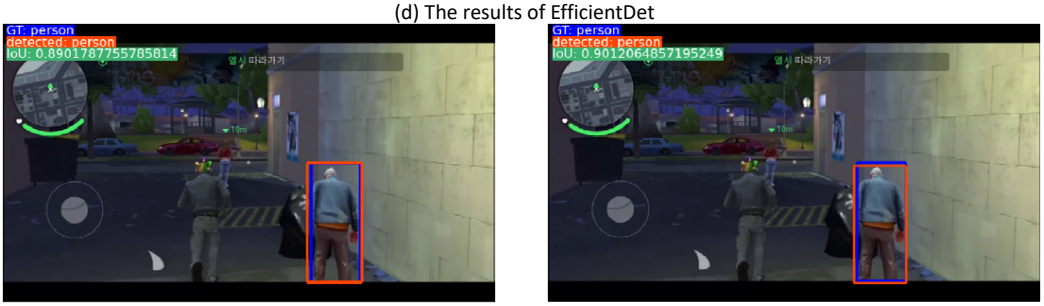
(b) The results of SSD



(c) The results of Faster R-CNN



(d) The results of EfficientDet



(e) The results of FPN

Figure A8. Comparison of the bounding box detection on game *Gangstar, New Orleans*. The left column is the result from the models trained by PASCAL VOC and the right column is the result from the models trained by PASCAL VOC and game scenes.

References

1. Redmon, J.; Farhardi, A. YOLOv3: An Incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
2. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
3. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A. SSD: Single shot multiBox detector. In Proceedings of the ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
4. Lin, T.-Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
5. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and Efficient Object Detection. In Proceedings of the 2020 Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10778–10787.
6. Everingham, M.; van Gool, L.; Williams, C.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
7. Everingham, M.; Eslami, S.A.; van Gool, L.; Williams, C.; Winn, J.; Zisserman, A. The Pascal Visual Object Chasses Challenges: A Retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
8. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollar, P.; Zitnick, C. Microsoft COCO: Common Objects in Context. In Proceedings of the ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
9. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), San Diego, CA, USA, 20–26 June 2005; pp. 886–893.
10. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
11. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
12. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014), Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
13. Girschick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV 2015), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
14. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. . [CrossRef] [PubMed]
15. Utsumi, O.; Miura, K.; Ide, I.; Sakai, S.; Tanaka, H. An object detection method for describing soccer games from video. In Proceedings of the IEEE International Conference on Multimedia and Expo, Lausanne, Switzerland, 26–29 August 2002; pp. 45–48.
16. Chen, Z.; Yi, D. The Game Imitation: Deep Supervised Convolutional Networks for Quick Video Game AI. *arXiv* **2017**, arXiv:1702.05663.
17. Sundareson, P. Parallel image pre-processing for in-game object classification. In Proceedings of the IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Bengaluru, India, 5–7 October 2017; pp. 115–116.
18. Venkatesh, A. Object Tracking in Games Using Convolutional Neutral Networks. Master's Thesis, California Polytechnic State University, San Luis Obispo, CA, USA, 2018.
19. Liu, S.; Zheng, B.; Zhao, Y.; Guo, B. Game robot's vision based on faster R-CNN. In Proceedings of the Chinese Automation Congress (CAC) 2018, Xi'an, China, 30 November–2 December 2018; pp. 2472–2476.
20. Chen, Y.; Huang, W.; He, S.; Sun, Y. A Long-time multi-object tracking method for football game analysis. In Proceedings of the Photonics & Electromagnetics Research Symposium-Fall 2019, Xiamen, China, 17–20 December 2019; pp. 440–442.
21. Tolmacheva, A.; Ogurcov, D.; Dorrer, M. Puck tracking system for aerohockey game with YOLO2. *J. Phys. Conf. Ser.* **2019**, *1399*. [CrossRef]
22. Yao, W.; Sun, Z.; Chen, X. Understanding video content: Efficient hero detection and recognition for the game Honor of Kings. *arXiv* **2019**, arXiv:1907.07854.
23. Spijkerman, R.; van der Haar, D. Video footage highlight detection in Formula 1 through vehicle recognition with faster R-CNN trained on game footage. In Proceedings of the International Conference on Computer Vision and Graphics 2020, Warsaw, Poland, 14–16 September 2020; pp. 176–187.
24. Kim, K.; Kim, S.; Shchur, D. A UAS-based work zone safety monitoring system by integrating internal traffic control plan (ITCP) and automated object detection in game engine environment. *Autom. Constr.* **2021**, *128*. [CrossRef]
25. YOLO in Game Object Detection. 2019. Available online: https://forum.unity.com/threads/yolo-in-game-object-detection-deep-learning.643240/ (accessed on 10 March 2019).
26. List of Video Game Genres. Available online: https://en.wikipedia.org/wiki/List_of_video_game_genres (accessed on 17 September 2021).
27. 7 Days to Die. Available online: https://7daystodie.com/ (accessed on 5 August 2013).
28. Left 4 Dead 2. Available online: https://www.l4d.com/blog/ (accessed on 17 November 2009).

29. Gangstar New Orleans. Available online: https://www.gameloft.com/en/game/gangstar-new-orleans/ (accessed on 7 February 2017).
30. Sims4. Available online: https://www.ea.com/games/the-sims/the-sims-4 (accessed on 2 September 2014).
31. Animal Crossing. Available online: https://animal-crossing.com/ (accessed on 14 April 2001).
32. Doraemon. Available online: https://store.steampowered.com/app/965230/DORAEMON_STORY_OF_SEASONS/ (accessed on 13 June 2019).
33. Asphalt 8. Available online: https://www.gameloft.com/asphalt8/ (accessed on 22 August 2013).
34. FIFA 20. Available online: https://www.ea.com/games/fifa/fifa-20 (accessed on 24 September 2019).
35. Huang, S.; Ramanan, D. Expecting the Unexpected: Training Detectors for Unusual Pedestrians with Adversarial Imposters. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 2243–2252.
36. Chan, Z.; Ouyang, W.; Liu, T.; Tao, D. A Shape Transformation-based Dataset Augmentation Framework for Pedestrian Detection. *Int. J. Comput. Vis.* **2021**, *129*, 1121–1138. [CrossRef]
37. Kang, H.; Lee, S.; Chui, C. Flow-based image abstraction. *IEEE Trans. Vis. Comp. Graph.* **2009**, *15*, 62–76. [CrossRef] [PubMed]
38. Winnemoller, H.; Olsen, S.; Gooch, B. Real-time video abstraction. *ACM Trans. Graph.* **2006**, *25*, 1221–1226. [CrossRef]
39. Kim, J.; Kim, M.; Kang, H.; Lee, K. U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. In Proceedings of the 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.