Semi-automatic Learning Framework Combining Object Detection and Background Subtraction

Sugino Nicolas Alejandro[1], Tsubasa Minematsu[1], Atsushi Shimada[1], Takashi Shibata[2], Rin-ichiro Taniguchi[1], Eiji Kaneko[2] and Hiroyoshi Miyano[2]

[1]*Graduate School of Information Science and Electrical Engineering, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan*

[2]*Data Science Laboratories, NEC Corporation, 1753 Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan*

Keywords: Object Detecion, Change Detection, Background Subtraction, Incremental Learning, Fine Tuning.

Abstract: Public datasets used to train modern object detection models do not contain all the object classes appearing in real-world surveillance scenes. Even if they appear, they might be vastly different. Therefore, object detectors implemented in the real world must accommodate unknown objects and adapt to the scene. We implemented a framework that combines background subtraction and unknown object detection to improve the pretrained detector's performance and apply human intervention to review the detected objects to minimize the latent risk of introducing wrongly labeled samples to the training. The proposed system enhanced the original YOLOv3 object detector performance in almost all the metrics analyzed, and managed to incorporate new classes without losing previous training information.

1 INTRODUCTION

The computer vision field was revolutionized by the introduction of convolutional neural networks (CNNs) and the impressive object detection performance achieved by the most recent proposed models such as the faster R-CNN (Ren et al., 2015), YOLO (Redmon and Farhadi, 2018), and Retinanet (Lin et al., 2017). To train these networks, large datasets are labeled manually and widely available (PASCAL-VOC (Everingham et al., 2010) and MS-COCO (Lin et al., 2014)). However, these public datasets do not contain new object classes and new forms of classes appearing in real-world surveillance scenes. Therefore, detectors should accommodate unknown objects and adapt to the target surveillance scene.

Many researchers have attempted to improve object detectors based on fine tuning (Yu et al., 2019a) (Mhalla et al., 2017). Fine-tuning-based methods extract candidates of known object classes using pretrained detectors and background subtraction methods and then re-train a pretrained detector using these candidates. The retrained detector forgets the prior knowledge possessed by the pretrained detector. In (Bendale and Boult, 2015), unknown objects are learned based on open set recognition task. We are inspired by (Bendale and Boult, 2015); thus, we propose a general learning framework using a change detection method to accommodate unknown objects.

In our framework, we employ background subtraction to detect unknown objects for the pretrained detector. If we use the detected areas as training samples based on a fully automatic labeling strategy, a latent risk of introducing wrongly labeled samples exists. Hence, we apply human intervention to filter the best samples and create a curated dataset for retraining.

We designed a system that performs object detection on a YOLOv3 CNN with a background subtraction algorithm and then extracts the detected objects and moving areas from the images to cluster them for easy manual labeling. Finally, a mix of the human-revised labeled data and the original data is used to improve YOLO's performance in detecting objects different from those contained in the original training dataset (fine tuning) and also to add new detection classes (incremental learning).

The main contribution of our study is the proposal of a framework that combines change detection and object detection to perform semi-automatic training. Furthermore, we implemented the system using a state-of-the-art object detector combined with tra-

Alejandro, S., Minematsu, T., Shimada, A., Shibata, T., Taniguchi, R., Kaneko, E. and Miyano, H.
Semi-automatic Learning Framework Combining Object Detection and Background Subtraction.
DOI: 10.5220/0008941200960106
In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2020) - Volume 5: VISAPP*, pages 96-106
ISBN: 978-989-758-402-2; ISSN: 2184-4321

ditional background subtraction methods, in addition to simple but fast object tracking and clustering techniques. Finally, we provide quantitative and qualitative results that demonstrate the increase in detection performance and the detection of new classes, respectively.

2 RELATED STUDIES

2.1 Open Set Problem for Pattern Recognition

In many actual scenarios for vision applications such as visual surveillance, a trained recognition system should accommodate unknown objects. This task is called as "open set recognition". The importance of this task was reported by (Scheirer et al., 2012). Inspired by recent progresses in pattern recognition, the open set recognition task was reorganized by (Bendale and Boult, 2015).

A critical and challenging issue on open set problems is the identification of unknown objects and definition of new objects. To address this issue, (Bendale and Boult, 2016) proposed the OpenMAX architecture, which is effective for identifying new objects. Asking a human about unknown objects is effective for defining unknown objects. For example, Uehara et al. proposed a framework for identifying and defining unknown classes using visual question generation (Uehara et al., 2018). Additionally, time-series videos are useful for identifying unknown objects and defining their classes by clustering the objects extracted from stereo with tracking information (Osep et al., 2019). In our proposed method, object-detection- and background-subtraction-based approaches are employed to identify unknown objects and define unknown classes.

2.2 Background Subtraction Methods to Complement Object Detection

Combining background subtraction methods with object detection was explored by (Rosenberg et al., 2005) where a semi-supervised training method that uses a background model is used to define the probability of an object or clutter to then retrain the detector model with the best samples. In (Yu et al., 2019a), background subtraction is applied to extract all the foreground objects in a specific scene and fine tune a detector to a certain class, thereby creating an anomaly detector. To reduce computational workload, (Wei and Kehtarnavaz, 2019) used a frame difference background subtraction model to propose areas of interest and passed them to a more robust CNN-based classifier to detect persons carrying specific objects in surveillance data captured from a long distance. Likewise, (Yousif et al., 2018) utilized a dynamic background model to develop region proposals to optimize a CNN and classify those regions within *human*, *animal*, and *background* categories. Meanwhile, (Yu et al., 2019b) used the output from the first layer of a CNN to extract features to replace the input of some traditional background subtraction methods and achieved improved performance with dynamic background scenes.

To accommodate the possibility of introducing wrong labels owing to automatic labeling, we rely on human intervention to filter and correct the dataset. In addition, we propose clustering in different dimensions to reduce the laborious work involved.

2.3 Scene-specific Training

Scene-specific training for distinct classes is crucial when implementing a generic detector (trained with widely available datasets) in real-life situations. In (Hammami. et al., 2019) the problem is mitigated using a generic detector to propose the detection of samples; subsequently, target scene object trajectories are calculated and associated with feature vectors extracted from the detected samples using a CNN. Using the corrected best samples, a new dataset is created and used to fine tune a detector. In addition, (Namatevs et al., 2019) proposed a YOLO CNN to detect objects that cross a virtual detection line and accelerate the training of an RNN (Recurrent Neural Network) with long short-term memory; thus, a neural network to generate automatic data labels for another network's training is used. (Mhalla et al., 2017) and (Maamatou et al., 2016) fine-tuned an object detector using samples generated by a generic detector, where a sampling step was applied to select the best candidates to build a specialized dataset. (Mhalla et al., 2017) utilized a likelihood function to filter the samples based on a background algorithm detection.

Unlike the aforementioned methods, our method relies on background subtraction to detect moving areas and create objects out of them, while maintaining the object detector output. Hence, the system can detect any type of moving object in the scene, regardless of the CNN's trained class set and the original ones whether they are moving or not.

3 PROPOSED FRAMEWORK

The main goal of the system is to improve the performance of an already trained object classifier relying on human work in a semi-automatic manner to reduce the laborious work involved. We propose a system that comprises three main modules: candidate extractor, clustering, and retraining. The system workflow can be seen in Figure 1.

3.1 Candidate Extractor

To improve the object classifier performance, objects undetected by the original network should be detected. We used two algorithms that perform in parallel, in which the classification network performance can be improved by retraining it with new annotated data, and a change detection method that performs foreground segmentation, which will be a basis for creating new classes or generating more data of an already known class.

Because the focus of the current study is on fixed point camera videos, we consider the change detection method as a good option because typically, the objects of interest in a scene will move or be moved. Currently, object classifiers are excellent at recognizing objects in an image; however, if they are not trained for a specific class or if their training samples do not contain a particular case of an object, the network will not classify the object or will classify incorrectly.

We will call the *candidate* to every detected object by the object classifier or independent blob after background segmentation. The purpose of this module is to extract candidates such that a user can review them and correct or add classifications if required. We can classify three types of candidates that we wish to extract:

- Correctly classified candidates: We do not want to lose this information.
- Incorrectly classified candidates: The original label must be corrected.
- Unclassified candidates: If they are object of interest to the user, they must be classified.

3.2 Clustering

The sum of all the candidates extracted by both the object classifier and the change detection method in each frame for a period of time results in many candidates to be manually labeled by a user. Hence, a clustering module is required to automatically group the same appearances of an object in a video or similar objects that might be of the same class, thus finally reducing the time and difficulty of the human intervention to add or correct the labels.

In the proposed framework, the clustering process is performed in two stages: 1) temporal grouping, and 2) grouping based on features and labels.

The first criterion for clustering is time and position, where objects of similar size in the same or close positions in consequent frames are considered to be the same. This is the first method in which we can enhance the network performance; for example, an object can be classified by the object detector in one frame but not in the next one owing to noise or occlusion. If the background subtraction module detects this object, both objects will be grouped together and share the label.

Subsequently, if the objects share a label, they will be clustered together. Finally, groups that are left behind with no labels are clustered by similarity of object color and shape. These last resulting clusters are left unlabeled and the reviewing is left to the user.

3.3 Manual Intervention and Incremental Learning

To enhance the performance of the original object detection, new and correct label information is required. To acquire this, we leverage human intervention. In addition, fine tuning or incremental learning is required to adjust the network for the new occurrences of known classes or to add new classes to the network, respectively.

All the previously mentioned steps were performed to minimize human intervention. Two primary steps are involved: first, filtering objects that do not belong to a certain cluster and reviewing the proposed clusters; next, input labels for the curated clusters and/or orphan groups of images that were not clustered. Both steps can be performed with minimal domain knowledge.

After the user review, the system takes the final clusters with their labels and automatically annotates the previously captured images in which the candidates appear. The retraining configuration will vary depending on the type of object in the clusters; if they are already known classes, we can add the new data and continue the original training with the previous network configuration. However, if a new class appears, we have first to modify the network architecture and then perform the retraining.

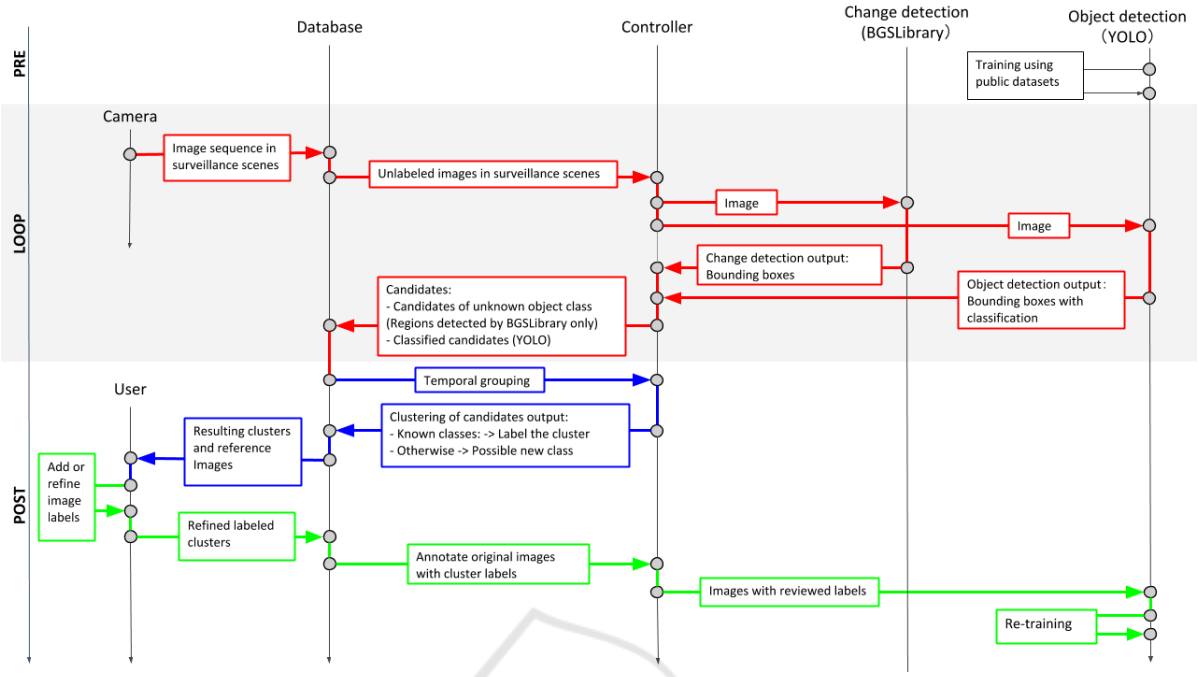

Figure 1: Proposed system overview. The *Candidate Extractor* (modules in red) is composed by the change detection and object detection modules. The *Clustering* (modules in blue) is performed in two stages: first, a temporal grouping and then based on features and labels. The *Re-training* (modules in green) comprises the user interaction that is required to add or refine labels for the automatically proposed clusters and the fine-tuning or incremental learning performed to the detector with the curated dataset.

4 IMPLEMENTATION

4.1 Candidate Extractor

The candidate extractor combines two modules: an object detector, YOLOv3 CNN, and a traditional background subtraction method (BGS), either PAWCS(St-Charles et al., 2016) or SuBSENSE (St-Charles et al., 2015). Both modules are fed with video frames in sequence from a camera or a file. YOLO's output are the detected object's bounding boxes, their classification, and a confidence ratio (classifications under a configurable confidence threshold are discarded). Meanwhile, BGS's output is a frame that is conformed by activated pixels where a foreground object had moved. We find contours in that frame, and retain only the extreme outer ones. Finally, we create bounding boxes for the closed contours and identify them as detected objects, such that YOLO's output can be compared.

An example of the output of YOLOv3 and the two background subtraction methods is shown in Figure 2. This image shows some cases of interest for the *Candidate Extractor*: As shown in the figure, the person is recognized by both the YOLO and the BGS methods; however, the sofa and chair are only detected in YOLO (because they are part of the background), whereas the box is only detected by the BGS module (YOLO was not trained for the box class). All these different cases had to be considered when designing this module such that the network does not lose previous knowledge and can incorporate new ones.

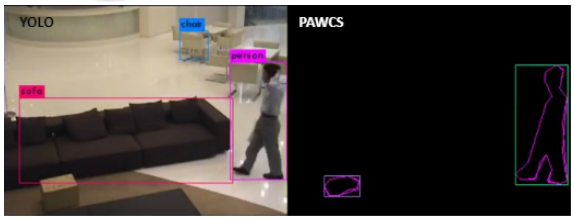

Figure 2: Debug output example of the *sofa* scene: Left: YOLOv3 (threshold:0.5). Right: PAWCS. The box is detected only by PAWCS, the sofa and chair only by YOLO.

4.2 Grouping and Clustering Module

We first implemented the temporal grouping, which is represented by groups; each *groups*, is conformed by images of an object that has been found in consequent frames (either by YOLO or BGS modules). If an ob-

ject does not remain in scene for a minimum number of frames, it is considered as noise and discarded. Figure 3 illustrates how a group is formed using a simple object tracking algorithm based on the size and position of the detected bounding boxes. If a bounding box is detected in consequent frames with a size and position difference within a threshold, it is considered the same; otherwise, it is considered as noise. This threshold might need to be adjusted if the refresh rate of the camera varies or if the objects moved faster than expected. As BGS's bounding boxes were more sensitive to illumination changes and shadows, if both modules detect an object, the system will always prioritize YOLO's bounding boxes.

Next, the *groups* are bundled in even larger clusters; therefore, only one label is required for all the images inside of them, thus reducing human labor. We define these larger bundles as *clusters* .The final output of this module comprises two different types of *clusters*:

- Labeled clusters: Groups that have the YOLO classification (e.g. car, boat).
- Unlabeled clusters: Groups who do not have classification and whose color histogram is *similar*.

4.2.1 Labeled Clusters

To prioritize YOLO classified objects, we first check within a *group* for classified candidates and then retain the most frequent classification to label the entire *cluster*. Within a labeled cluster, wrong classifications may appear or an entire *cluster* might be wrong. Hence, by reviewing and retraining the network, we can increase its true positives ratio.

4.2.2 Unlabeled Clusters

To create the unlabeled clusters, we first use some images samples from each *group* that had no classification and convert the RGB captured images into the HSV colorspace. We use more than one candidate from each *group* because even though the *group* consider the temporal and spatial information, we could not guarantee that the candidates in each *group* represented the same object, as a simple object tracking was implemented to group them. HSV colorspace was selected because it separates the luminescence; therefore, the candidates become more robust to lighting changes.

We compute the histograms of the three candidates of each group; subsequently, they are used to calculate a distance matrix with the average Bhattacharyya distance between the candidates of a group and the ones from the others. This distance metric provided the best results based on a qualitatively comparison between Correlation, Chi-Squared, and Intersection distances.

Finally, DBScan (Ester et al., 1996) clustering algorithm is used with the precalculated distance matrix. This algorithm requires two parameters: ε, which is the maximum distance between two objects, and $minPts$, which is the minimum number of neighbors required in a sample for it to be considered a core. These parameters must be adjusted manually; the system will perform clustering with default values and the user will adjust the parameters according to the resulting clusters and noise samples. DBScan was selected because it does not require previous information regarding the number of clusters.

Examples of the unlabeled clusters and noise points are shown in Figure 4. Some clusters might not be of interest and some groups might be misplaced or included as noise points. Hence, manual intervention is necessitated in the next step.

4.3 Manual Intervention and Re-training

In addition to the misplacements in clusters that may occur with unlabeled clusters, YOLO may misclassify objects. Manual intervention is required to adjust the labels, remove *groups* out of a *cluster* in which they do not belong, and label unlabeled clusters. Furthermore, the user might select individual groups from the unclustered group (noise points) or those that were removed from their clusters, and then label them correctly. Figure 5 shows the process that must be performed in this step.

After the user reviewing and labeling, the module searches for the original images where all the candidates in each group objects appear and generates the annotation data using the bounding box information previously generated by the *Candidate Extractor*, thus generating an output dataset ready for retraining the CNN. Figure 6 illustrates this automatic process.

For the retraining, we use the annotated data after human review and combine it with the original COCO dataset. We retain the original trained weights of the network as the initial weights for retraining because the goal is to enhance the default performance for the scene in the study; subsequently, starting from those weights, we perform retraining until the desired performance is achieved. When adding new classes to the network, the latter's last layers must be modified. and neurons must be added for classifying the new classes in the last layers.

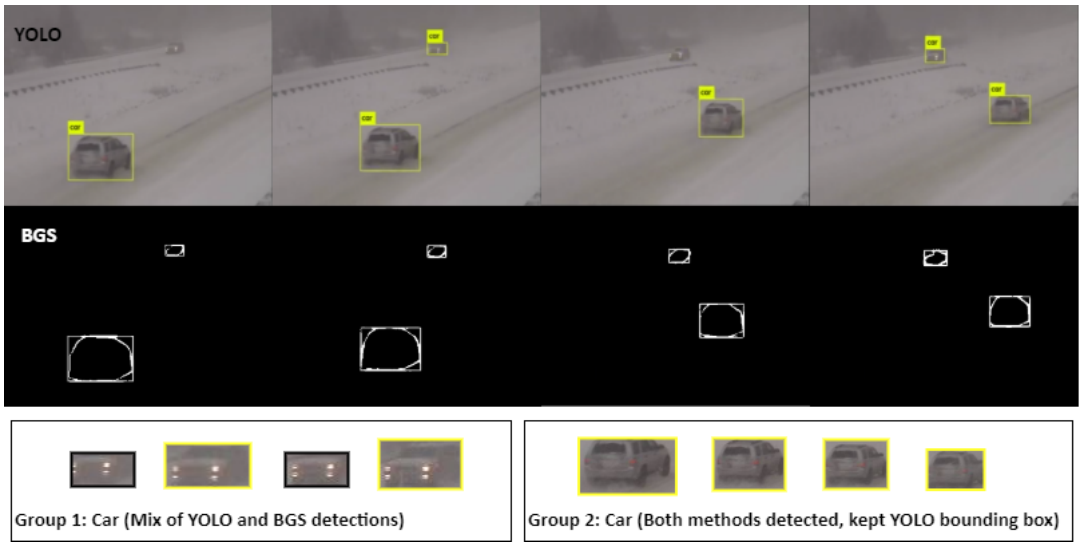

Figure 3: Candidates and groups extracted from *blizzard* scene. Up: YOLO detections, Middle: BGS detections contour and bounding boxes. Down: Two output *groups*, black border candidates are extracted from SuBSENSE, the yellow border ones from YOLO.

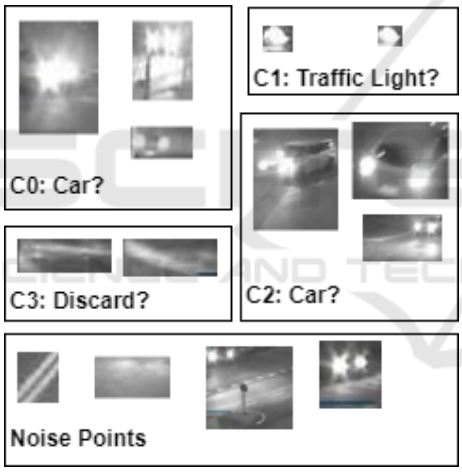

Figure 4: Unlabeled clusters from *street* scene. C0, C1, C2, C3: Unlabeled clusters with their possible true class.

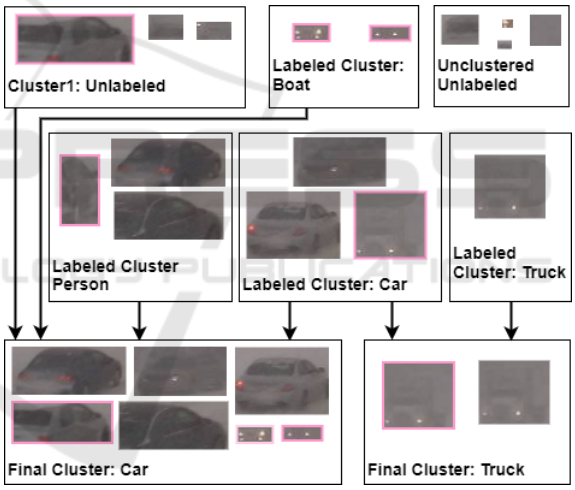

Figure 5: Diagram of the selection process from *blizzard* scene. Highlighted *groups* do not belong to the true class and must be corrected manually. Up: Output from clustering, labeled and unlabeled *clusters*. Down Cluster: User-reviewed clusters with *car* and *truck* labels.

5 EXPERIMENTS

Two types of experiments were performed for testing the entire system, the goal of which is to enhance the performance of an already trained network, i.e., YOLOv3 with its default weights, in a certain condition or type of unknown object:

- Train to improve the detection of an already trained class.
- Train to detect a new class.

All the experiments were performed using the CD.net 2014 (Wang et al., 2014) dataset that is primarily surveillance videos from fixed point cameras, the selected scenes were split in two, using the first half for training and the second one for validation. The hardware used was a server equipped with Intel(R) Xeon(R) CPU E5-2686 v4 and an NVIDIA Tesla K80. YOLOv3 was used with the Darknet framework from its original creator using CUDA acceleration, both for original object detection and retraining. For the background subtraction methods, BGSLibrary's (Sobral and Bouwmans, 2014) implementation of various algorithms was used.

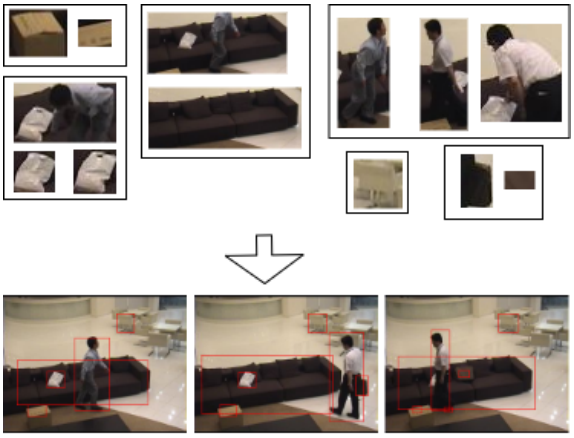

Figure 6: Diagram of the automatic annotation after manual labeling of clusters selection process from MS-COCO *sofa* scene. Upper groups: Reviewed clusters. Down: Annotated data (red squares drawn only for ease of understanding).

5.1 Selection of a Background Subtraction Method

We reviewed traditional background subtraction algorithms and assessed their performances against YOLO. Based on the information provided by previous CD.net 2014 results and the algorithms available in BGSLibrary, a test suite was prepared to compare several algorithms and obtain quantitative results to select the best fit.

To select the best method, we primarily considered a high recall and F-measure as after the detection, the results will be reviewed by a human aided by a clustering method. We selected SuBSENSE as our main background subtraction method and retained the possibility for using PAWCS as it demonstrated better performances in some specific scenes such as highly dynamic background ones. In addition, using its default parameters, PAWCS was less sensitive than SuBSENSE (lower recall) and produced more consistent blobs. Results of the selected methods are shown in Table 1; the metrics used are the same as those proposed in CD.net 2014 and were measured using the provided tools.

Table 1: Results of some of the tested BGS methods on all sections averaged on 2014 CD.net dataset. *R*: Recall. *Sp*:Specificity. *P*: Precision. Diff: Frame Difference background subtraction, its results are provided for comparison.

	R	Sp	P	F-measure
Diff	0.36	0.95	0.38	0.23
PAWCS	0.72	0.98	**0.79**	0.69
SubSENSE	**0.78**	**0.99**	0.77	**0.73**

5.2 Improving Detection of an Already Trained Class

Based on the pre-experimental results, we discovered scenes where YOLO did not perform as expected. Scenes that exhibited issues were the following:

- Turbulence: No detection or wrong classification.
- Bad weather, night: Misclassifications, low recall.

From the original CD.net 2014 dataset, we selected *turbulence0* and *turbulence2* scenes from *turbulence* cases, the *blizzard* scene from *badWeather*, and *streetCornerAtNight* (referred as *street*) and *busyBoulevard* (referred as *boulevard*) from *night* scenes. We generated labels for validation using the ground truth data in some of the scenes, and used them to measure the quantitative results. Details of the training are shown in Table 2.

The manual intervention process involves discerning the resulting *clusters* and searching for those that we intend to use (either labeled or unlabeled ones); subsequently, we review them and filter out *groups* that did not belong to the cluster (e.g., in Figure, 5 we remove the *truck* from the car class). Finally, we assign labels to the final *clusters*. In some cases we reviewed the orphan *groups*, such as the noise points and those filtered out from a cluster, and we assigned a label to them (e.g., in Figure, 5 after filtering out the *group* that has a truck and label it as *truck*).

We fine-tuned the network starting from the original YOLOv3 weights, with no added classes. The training parameters are the following (Unless otherwise noted, the YOLOv3 default training configurationis used hereinafter): Using $width = height = 416$, the training duration is specified in iterations (Table 3 and 4), each iteration is a full batch training (64 images). Each batch is composed of 63 images from MS-COCO and one image from the new dataset generated after the manual labeling. The MS-COCO dataset is composed of 117263 images for training and 5000 for validation. Depending on the scene, a few thousand images are added to both datasets (Table 2).

Some detection parameters had to be adjusted for each scene in particular. These parameters affect on how well the cluster module performs and thus how much easier the labeling becomes. These parameters are: the background subtraction method (SuBSENSE or PAWCS), and the external parameters required for DBScan (maximum distance and minimum points).

To determine when the training is completed, the loss function was ineffective as it did not provide much information, as the network, i.e., original YOLOv3, has already been trained for over 500000

Table 2: Frames used for re-training the network. GT: Has Ground Truth labels. The training and validation columns refer to the starting and last frames from the scene. Clusters shows the total clusters and how they are composed $Unlabeled + Labeled$.

Scene	GT	BGS Method	Training	Validation	Detections	Groups	Clusters
turbulence0	Yes	SuBSENSE	1324-1989	1990-2593	1909	21	6(2+4)
blizzard	Yes	SuBSENSE	906-1819	1820-3587	4235	67	5(1+4)
street	Yes	PAWCS	862-1621	1622-2999	2520	97	9(5+4)
turbulence2	No	SuBSENSE	59-2733	2734-4397	2752	11	1(1+0)
boulevard	No	SuBSENSE	755-1290	1291-1829	10951	438	7(3+4)

Figure 7: Frames from *street* scene. Left: Car wa only detected by BGS. Right: Car detected by YOLO.

iterations using the MS-COCO dataset. Therefore, the networks were trained until satisfactory measures were achieved using the ground truth semi-automatic labels or with qualitative evaluation depending on the case.

We measured the original YOLOv3's and the retrained network's performance using the same test bench so as to keep results comparable. Considering TP = Number of true positives, FP = number of false positives, FN = number of false negatives, P = Precision, R = Recall, F = F-measure and IoU = Average Intersection over Union, all the results can be seen in Table 3.

In all the studied cases the network performance was enhanced for the specific scene. Enhancements of Precision, Recall, and thus F-Measure are shown in almost all scenes with the different thresholds analysed. The most evident case is the *turbulence0* scene were the original YOLOv3 was almost not able to detect anything correctly. The *street* scene at 0.25 threshold measurements showed some deterioration compared to the original YOLOv3. After a thorough review of the images, we discovered that this was due to the background subtraction method that only detected the lights of cars at night and not the full car; therefore, the bounding boxes deviated slightly from the expectation, reducing both the Precision and IoU, as shown in Figure 7.

In addition to the quantitative results obtained from the ground truth labels, we computed the performances of the retrained networks with the original COCO dataset; the results are shown in Table 4. Furthermore, we added $mAP@IoU = 0.5$ (mean Average Precision) to the previous measurements. i.e., the mean average precision over all 80 MS-COCO categories where the IoU of a TP is over 50%, which is typically used to compare the accuracy of detection algorithms in object detection. Fine-tuning usually implies losing previous knowledge, the results shows that we did not lose previous knowledge significantly. The slight decrease in Recall is compensated with a higher Precision resulting in similar F-Measure.

5.3 Training to Detect New Classes

Compared with the previous experiment, the only difference in the labeling process is that the user has to add new classes. Meanwhile, the training process is slightly different. In this case, because the number of objects to classify is different from that of the original network, we remove the last layers of YOLOv3, where the detection at three scales start. This implies using the weights up to the 81st layer and then retraining from thereof with the new structure that supports the new objects. Additionally, to determine when training is to be halted, we examined the loss function and used the weights from the iteration where no further decrease was shown.

For this case, we used the *sofa* scene of the *intermittentObjectMotion* cases. In this scene, the objects of interest are a box, plastic bag, and suitcase, which are placed on a sofa for a few seconds and then removed. The original YOLOv3 can recognize the suitcase in some frames; however, a class that represents the box or plastic bag does not exist. Both are objects of interest for this scene and are in fact moved; therefore, the BGS methods can identify them. As such, the total number of classes was 82 including the two new ones: *box* and *plastic bag*.

In this scene, we prefiltered the training images before retraining. To reduce the risk of catastrophic forgetting to the minimum, we defined the sofa and chair as *background objects* such that if any of them was not labeled in an image, it was not used for training. This is because YOLO detected *chair* and *sofa* in many frames but not in all of them (1480 out of 2692). Because 1480 frames were considered enough for training, we could discard those frames and use those fully labeled for those classes.

All the frames of the video, after the prefiltering of the best samples, were input to the system. The

Table 3: Performance of the different trained networks based on the ground truth labels. turb0 refers to *turbulence0* scene.

Scene	Network	Iterations	threshold	P	R	F	TP	FP	FN	IoU[%]
street	yolov3	-	0.1	0.59	0.29	0.33	114	79	375	45
	proposed	9000	0.1	0.71	**0.46**	**0.56**	380	154	454	52.11
	yolov3	-	0.25	**0.89**	0.19	0.31	93	11	396	**68**
	proposed	9000	0.25	0.88	0.31	0.46	259	35	575	66.86
turb0	yolov3	-	0.1	0	0	0	2	1547	1285	0
	proposed	3000	0.1	0.54	**0.75**	0.63	969	813	318	39
	yolov3	-	0.25	0	0	0	0	655	1287	0
	proposed	3000	0.25	**0.81**	0.69	**0.75**	894	214	393	**58.24**
blizzard	yolov3	-	0.1	0.69	0.62	0.65	1050	469	657	58
	proposed	6000	0.1	0.83	**0.7**	0.76	1201	248	506	70.42
	yolov3	-	0.25	0.87	0.58	0.7	994	151	713	73
	proposed	6000	0.25	**0.95**	0.66	**0.78**	1201	62	586	**80.94**

Table 4: Performance of the different networks on MS-COCO dataset. Measured yolov3 original network is included for reference. turb0 refers to *turbulence0* scene, It: Iterations, thre: Detection threshold, mAP: mean Average Precision.

Scene	Network	It	thre	P	R	F	TP	FP	FN	IoU[%]	mAP[%]
-	yolov3	-	0.25	0.64	0.54	0.59	19320	10879	16437	50.5	54.65
street	proposed	9000	0.25	0.66	0.51	0.58	18308	9316	17449	52.37	53.6
turb0	proposed	3000	0.25	0.64	0.5	0.57	18049	9993	17708	50.58	51.89
blizzard	proposed	6000	0.25	0.66	0.51	0.58	18255	9382	17502	52.17	52.17
-	yolov3	-	0.1	0.47	0.62	0.53	22020	24993	13737	36.23	54.64
street	proposed	9000	0.1	0.48	0.6	0.54	21616	23214	14141	37.48	53.63
turb0	proposed	3000	0.1	0.46	0.6	0.52	21312	24827	14445	32.72	51.89
blizzard	proposed	6000	0.1	0.48	0.6	0.53	21547	23341	14210	37.3	53.35

captured frames from the resulting video where these cases appear are shown in Figure 8, with the original YOLOv3 as a reference. The retrained network can detect and classify the box, plastic bag, and suitcase correctly. The box on the sofa is not shown in the images as labeled but was classified at a lower confidence ratio (15%). This is because the box over the sofa was not part of the training dataset but rather of the validation dataset.

5.4 Qualitative Evaluation

Some of the scenes mentioned in the sections above have been annotated from their ground truth data to create metrics and quantitative results. The other scenes that were analyzed only qualitatively include *sofa*, *boulevard*, and *turbulence2*. The results are shown in Figures 8, 9 and 10 respectively, all these cases were classified with the same threshold (0.25).

The captions provide different cases of interest, such as fixing a YOLO label in the *sofa* with the bench classification changed to chair, which was the human input, as well as increasing the Recall shown in the *boulevard* scene with more detections of persons and cars. New classes were detected in the *sofa* scene such as the box and plastic bag.

6 CONCLUSIONS

We herein propose a system to enhance neural network performance with minimal human intervention. We combined YOLOv3, a fast real-time capable CNN, with traditional background subtraction methods that require no training and could be used in many different cases, to improve the detection performance of the network. In addition, we applied time, position, classification, and color histogram similarity clustering policies to ease the labeling of a new dataset for training.

The proposed system enhanced the original YOLOv3 performance in almost all the metrics analyzed for the tested scenes.

Our implementation did not consider catastrophic forgetting nor more modern tracking and clustering methods. We expect our framework to be improved if such methods were used and catastrophic forgetting considered.

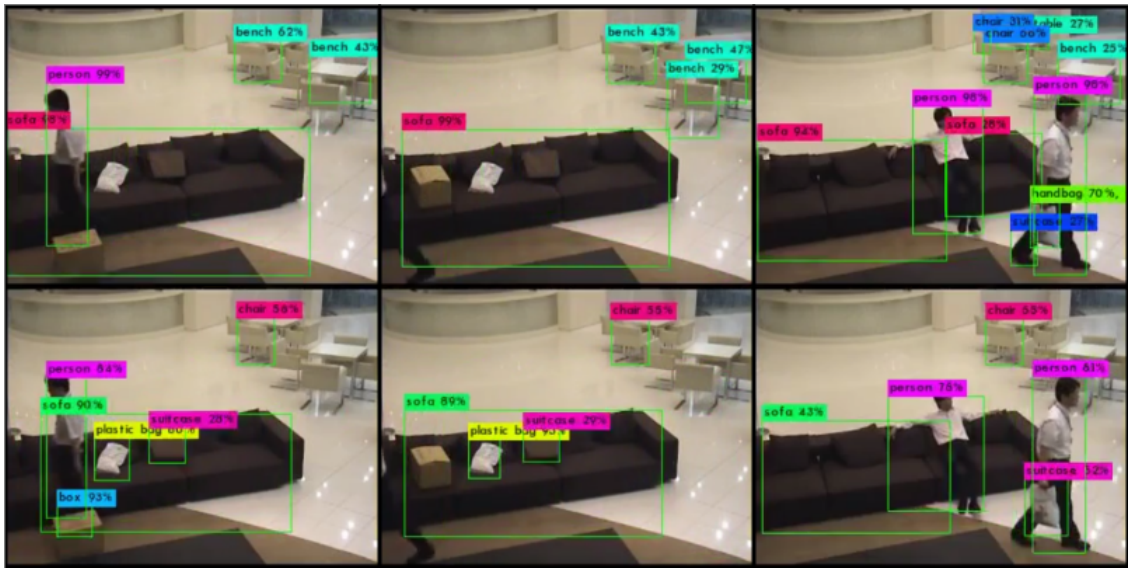

Figure 8: Captions from the *sofa* scene, Top row: Original YOLOv3 (threshold: 0.25), Down row: Retrained network after 7000 iterations (threshold: 0.25). Left column: Detection of all the objects of interest (*box*, *plastic bag*, *suitcase*). Middle column: FN of *box*. Right column: Improved detection of *sofa* and *suitcase*.

Figure 9: Sequences of *boulevard* frames before(up) and after(down) retraining for 10000 iterations, showing Recall increase in *car*(yellow) and *person*(purple).

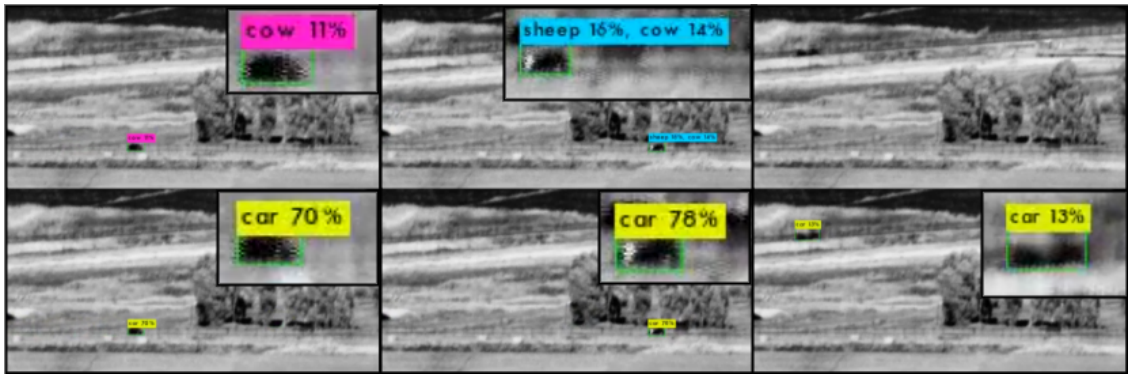

Figure 10: Sequences of *turbulence2* frames before(up) and after(down) retraining for 7000 iterations. Yellow labeled objects are correct detections of *car* (in yellow). Before retraining, the original network did not detect them or misclassified them as *cow* (magenta) or *sheep* (light blue).

REFERENCES

Bendale, A. and Boult, T. E. (2015). Towards open world recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1893–1902.

Bendale, A. and Boult, T. E. (2016). Towards open set deep networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press.

Everingham, M., Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338.

Hammami., N., Mhalla., A., and Landrault., A. (2019). Domain adaptation for pedestrian dcnn detector toward a specific scene and an embedded platform. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,*, pages 321–327. INSTICC, SciTePress.

Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. *CoRR*, abs/1708.02002.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

Maamatou, H., Chateau, T., Gazzah, S., Goyat, Y., and ESSOUKRI BEN AMARA, N. (2016). Transductive transfer learning to specialize a generic classifier towards a specific scene. pages 411–422.

Mhalla, A., Chateau, T., Mamatou, H., Gazzah, S., and Amara, N. E. B. (2017). Smc faster r-cnn: Toward a scene-specialized multi-object detector. *Computer Vision and Image Understanding*, 164:3 – 15. Deep Learning for Computer Vision.

Namatevs, I., Sudars, K., and Polaka, I. (2019). Automatic data labeling by neural networks for the counting of objects in videos. *Procedia Computer Science*, 149:151 – 158. ICTE in Transportation and Logistics 2018 (ICTE 2018).

Osep, A., Voigtlaender, P., Luiten, J., Breuers, S., and Leibe, B. (2019). Large-scale object mining for object discovery from unlabeled video. *arXiv preprint arXiv:1903.00362*.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks.

Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. *WACV/MOTION*, 2.

Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., and Boult, T. E. (2012). Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772.

Sobral, A. and Bouwmans, T. (2014). Bgs library: A library framework for algorithms evaluation in foreground/background segmentation. In *Background Modeling and Foreground Detection for Video Surveillance*. CRC Press, Taylor and Francis Group.

St-Charles, P., Bilodeau, G., and Bergevin, R. (2015). Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing*, 24(1):359–373.

St-Charles, P., Bilodeau, G., and Bergevin, R. (2016). Universal background subtraction using word consensus models. *IEEE Transactions on Image Processing*, 25(10):4768–4781.

Uehara, K., Tejero-De-Pablos, A., Ushiku, Y., and Harada, T. (2018). Visual question generation for class acquisition of unknown objects. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 481–496.

Wang, Y., Jodoin, P.-M., Porikli, F., Konrad, J., Benezeth, Y., and Ishwar, P. (2014). Cdnet 2014: An expanded change detection benchmark dataset. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 393–400, Washington, DC, USA. IEEE Computer Society.

Wei, H. and Kehtarnavaz, N. (2019). Semi-supervised faster rcnn-based person detection and load classification for far field video surveillance. *Machine Learning and Knowledge Extraction*, 1(3):756767.

Yousif, H., Yuan, J., Kays, R., and He, Z. (2018). Object detection from dynamic scene using joint background modeling and fast deep learning classification. *Journal of Visual Communication and Image Representation*, 55:802 – 815.

Yu, T., Yang, J., and Lu, W. (2019a). Combining background subtraction and convolutional neural network for anomaly detection in pumping-unit surveillance. 12:115.

Yu, T., Yang, J., and Lu, W. (2019b). Refinement of background-subtraction methods based on convolutional neural network features for dynamic background. *Algorithms*, 12:128.