electronics

Article

Key-Frame Extraction for Reducing Human Effort in Object Detection Training for Video Surveillance

Hagai R. Sinulingga and Seong G. Kong *

Department of Computer Engineering, Sejong University, Seoul 05006, Republic of Korea; hagairaja@sju.ac.kr
* Correspondence: skong@sejong.edu

Abstract: This paper presents a supervised learning scheme that employs key-frame extraction to enhance the performance of pre-trained deep learning models for object detection in surveillance videos. Developing supervised deep learning models requires a significant amount of annotated video frames as training data, which demands substantial human effort for preparation. Key frames, which encompass frames containing false negative or false positive objects, can introduce diversity into the training data and contribute to model improvements. Our proposed approach focuses on detecting false negatives by leveraging the motion information within video frames that contain the detected object region. Key-frame extraction significantly reduces the human effort involved in video frame extraction. We employ interactive labeling to annotate false negative video frames with accurate bounding boxes and labels. These annotated frames are then integrated with the existing training data to create a comprehensive training dataset for subsequent training cycles. Repeating the training cycles gradually improves the object detection performance of deep learning models to monitor a new environment. Experiment results demonstrate that the proposed learning approach improves the performance of the object detection model in a new operating environment, increasing the mean average precision (mAP@0.5) from 54% to 98%. Manual annotation of key frames is reduced by 81% through the proposed key-frame extraction method.

Keywords: object detection; video surveillance; key-frame extraction; interactive labeling; deep learning

Citation: Sinulingga, H.R.; Kong, S.G. Key-Frame Extraction for Reducing Human Effort in Object Detection Training for Video Surveillance. *Electronics* **2023**, *12*, 2956. https://doi.org/10.3390/electronics12132956

Academic Editor: Yu-Chen Hu

Received: 15 May 2023
Revised: 17 June 2023
Accepted: 3 July 2023
Published: 5 July 2023

1. Introduction

Video surveillance systems are widely used for detecting objects or events of interest in various applications, including evaluating traffic density [1], checking the safety of public buildings [2], and monitoring abnormal behaviors in cattle barns [3]. In traditional approaches, human operators must pay attention to video surveillance monitor screens 24/7 since events of interest can occur at any time. In cases, such as public safety, quickly detecting events of interest is essential to provide an early alert, ensuring prompt response.

Intelligent video surveillance systems utilize advances in artificial intelligence and computer vision techniques to achieve their stated goal automatically and autonomously. Deep learning models are often used as the basis for these systems, and typically require an initial step of identifying an object of interest to detect. This task involves encapsulating the object inside the video frame with a tight bounding box. Currently, the most accurate and real-time feasible approach is supervised learning-based deep learning algorithms, such as YOLO [4], which require labeled data of at least 1500 images per class, with each class instance appearing at least 10,000 times. Several public datasets, such as MS COCO [5] and ImageNet [6], are available to help create pre-trained YOLO models, which can extract common features, such as edges, textures, and contours, through the convolution process. However, these pre-trained models may not perform well when detecting objects in target surveillance videos that were not seen during training. Fine-tuning the pre-trained model with new data from a new surveillance video can help achieve the expected accuracy,

but the accuracy of the tuned model may significantly drop when deployed to monitor another scene.

There are several approaches to improve the annotation process, including weakly supervised learning, Mechanical Turk, and interactive learning. In weakly supervised learning [7], object detection is trained using image-level class labels instead of detailed bounding box annotations, simplifying and expediting the annotation process. However, this method is only suitable when there is a single object in the image. Mechanical Turk [6] is an annotation solution that outsources tasks to external parties, enabling faster annotation by involving a larger number of people. However, this approach may not be suitable for handling confidential data. On the other hand, interactive learning [8] combines human and AI efforts in the annotation process, reducing human workload by annotating only the data that AI fails to annotate accurately. While interactive learning has been successfully applied in image segmentation data preparation [9], its application in object detection remains relatively unexplored.

This paper presents a key-frame extraction method for collecting training images from a surveillance video stream to improve the performance of a pre-trained deep learning model in object detection. The proposed key-frame extraction scheme aims to effectively reduce false positives of the object detection model in a new operating environment. Key-frame extraction replaces human effort in screening important image frames for training from the new video stream to build custom datasets. This module compares object movement detection results using the optical flow algorithm [10] with the detection of the deep learning model. Static surveillance cameras operate on the same background, which is utilized by using optical flow because the object of interest will have movement in the video surveillance scene. The key-frame extraction is followed by interactive labeling, which labels the collected image frames with minimal human effort. Interactive labeling is inspired by the success of interactive learning in building models in domains that require an expert to annotate the data [11]. The proposed method uses an image classifier as a human annotator assistant in annotating object interests. The human annotator only needs to provide a bounding box with two points. The image is then cropped for each bounding box, and the image classifier predicts which class the object in the bounding box belongs to. The process is repeated several times until the retrained model achieves the desired performance. Experiment results show that the proposed scheme improves the performance of the object detection model in a new operating environment from 54% to 98%, measured via mAP@0.5, and reduces human effort by 81% compared to fully manual annotation of the key frames.

This paper presents three main contributions: (1) A key-frame extraction scheme to collect additional training images from surveillance videos to effectively reduce false negatives of the retrained object detection model in a new operating environment; (2) An interactive labeling scheme to annotate the collected images by incorporating an image classifier as a class labeling assistant, reducing human effort in the process; and (3) Evaluation of the proposed key-frame extraction and interactive labeling schemes through quantitative accuracy improvement experiments, applied to the YOLO real-time object detection model in surveillance videos.

2. Related Work

Inspired by the way the human eye works, CNN models [12] automatically find features in the image for object detection [13,14]. However, the drawback of deep learning methods is the requirement for a large amount of labeled data. The accuracy and consistency of annotations significantly impact the performance of these techniques. Therefore, extending them to scenarios with many classes would be labor-intensive.

To address this problem, researchers proposed weakly supervised learning for object detection [7]. The data needed to train the object detector based on deep learning is not a bounding box with a label, but only the label of the object in the image. This method generally uses two models, the pre-trained model and the target model, both of which have

the same architecture. The pre-trained model has been trained with massive datasets, such as COCO or ImageNet. This model can be a CNN-based model [15,16] or a transformer-based model [14]. The target model then tries to learn from the results of the pre-trained detection plus other modules. However, this method is limited to cases where there is only one salient object in the image.

Videos have a special characteristic where the contents of neighboring frames are similar. However, object detection in the video is not different from images, thanks to the development of the YOLO [17,18] and RCNN [19,20] methods, which achieve high accuracy and fast inference time. Thus, the remaining problem of how to scale the object detection algorithm is about efficiently obtaining annotated data with low human effort to feed the training of the object detection model. One way to address the problem is to outsource the data annotation task, such as through Mechanical Turk [6]. There is no need to design the best method for annotating data [21], including quality control [22]. However, this method is not an option when the data to be annotated is confidential, as in animal behavior or medical data.

Interactive learning [8] incorporates human-in-the-loop. This involves gradually annotating data so that the annotator does not need to put in too much effort until the model reaches the target accuracy. The method of selecting the next data to be annotated is based on the level of error generated in the prediction results. The image with the highest error is given priority for annotation for further learning. This approach is significant in the case of image segmentation [9], as annotating image segmentation takes a long time per image. A framework for interactive annotation [8] in object detection is less efficient as it checks all the images in the dataset.

3. Key-Frame Extraction for Object Detection Model Training

The additional training method incorporates the concepts of incremental and active learning. To improve the process of acquiring annotated data for fine-tuning object detection models, key-frame extraction is used. The key-frame filtering process consists of three sub-processes: motion detection (MD), object detection (OD), and false negative (FN) detection. The input to the motion detection is two consecutive frames (frames $X(t)$ and $X(t+1)$) from the target video stream, and the output is the bounding box of the moving object $B_{MD}(t)$ within the frame. The input to the object detection process is a frame $X(t)$, and the output is the bounding box of the object of interest $B_{OD}(t)$ within the frame, along with its class label. The FN detection process compares the intersection over union (IOU) between the results of $B_{MD}(t)$ and $B_{OD}(t)$ to determine whether $X(t)$ contains any FNs created via the object detection model. A video frame that contains at least one FN object is marked as X_{FN} for interactive labeling. Figure 1 shows the flow of the key-frame extraction process. During interactive labeling, the false negatives X_{FN} are labeled by both a human annotator and a trained image classifier (IC) working together to produce the ground-truth bounding box, denoted as B_{GT}.

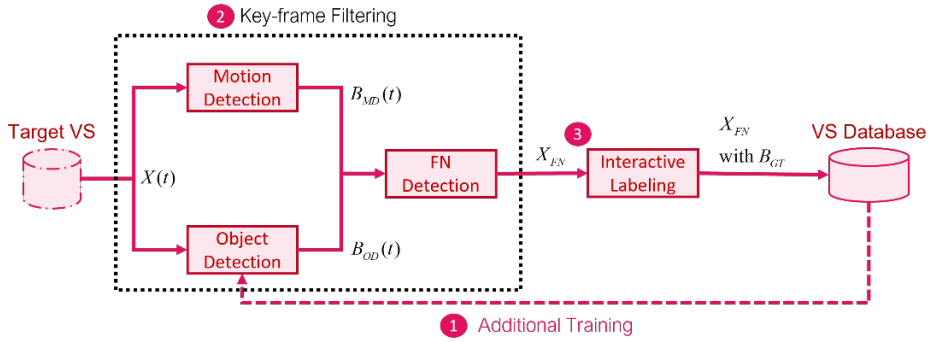

Figure 1. Overall diagram of key-frame extraction: (1) Additional training to improve the deep learning-based object detection model using data on the VS database, (2) key-frame filtering to collect the frames containing false negatives, and (3) interactive labeling for data annotation with less human effort.

3.1. Key-Frame Filtering

Key-frame filtering aims to replace human effort in finding key frames, which are frames in the video stream that have the potential to increase the accuracy of the object detection model. Key frames are collected to form a dataset with high variability to cover all possibilities that may occur later. This is a task that humans typically have to perform manually with a video stream. To find this data automatically, we propose using a pre-trained object detection model, such as YOLOv5, that has been tuned using the video stream database. This tuned model can detect parts of the object of interest, but there is still a high possibility that it may generate FP or FN results. FPs occur when the object detection model predicts a wrong detection, while FNs occur when the object detection model fails to find the object of interest. In this module, we focus on finding FNs by utilizing the characteristics of the object of interest that moves in the scene captured via the surveillance camera. We use dense optical flow (*OF*) [10] as the motion detector to compute $M(t)$, which includes the direction $M_d(t)$ and magnitude $M_m(t)$ of the movement of each pixel from two consecutive frames, specifically frames at time t and $(t + 1)$ seconds:

$$M_d(t) + M_m(t) = OF(X(t), X(t+1)) \tag{1}$$

The amount of 1 s depends on frame rate. For example, in a video stream with a frame rate of 15 fps (frames per second), the frames at t and $(t + 1)$ seconds correspond to frame #1 and #16, and #16 and #31, etc. The amount of movement per pixel is then filtered using a threshold θ_{Fil} to separate pixels with moving objects and those without. This process will generate a binary movement map $BM(t)$:

$$BM^{i,j}(t) = \begin{cases} 1 & \text{if } M_m^{i,j}(t) > \theta_{\text{Fil}} \\ 0 & \text{else} \end{cases} \tag{2}$$

With the binary map, we then use contour finding (*CF*) to localize moving parts that have the possibility of containing objects of interest $B_{MD}(t)$:

$$B_{MD}(t) = CF(BM(t)) \tag{3}$$

The *CF* function used is based on the implementation by [23], which can be called in Python 3 by invoking cv2.findContours from the OpenCV library. Figure 2 shows an example of the results of each step. To determine whether the frame obtained by *OF* is false negative or not, we compare it with the detection results from the object detection model

which is denoted as $B_{OD}(t)$. The comparison is conducted by calculating the IOU value between the two detection results:

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{B_{MD}(t) \cap B_{OD}(t)}{B_{MD}(t) \cup B_{OD}(t)} \quad (4)$$

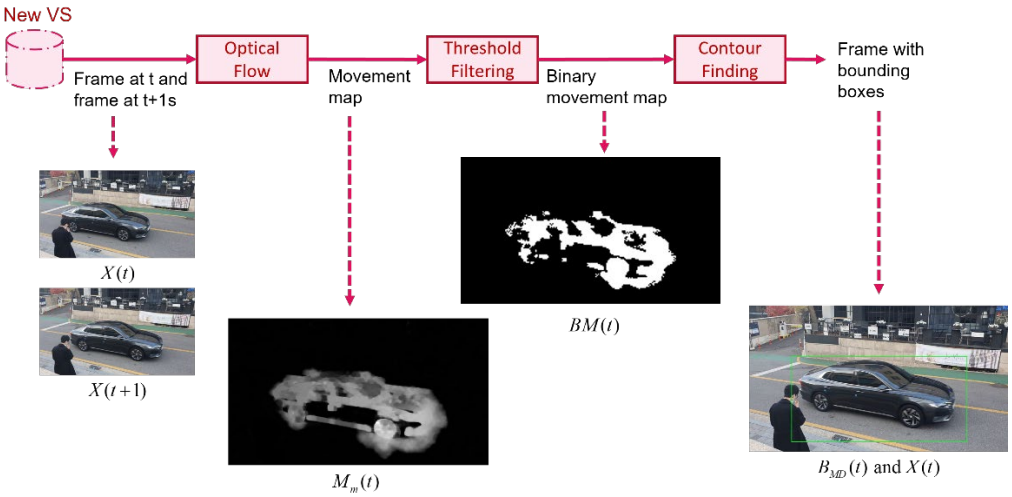

Figure 2. Steps of motion detector on filtering the frame from the target VS.

We assume that an FN occurs when a moving object is detected by MD but not by OD. There are four cases that may arise: (1) Both OD and MD detect the object; (2) OD detects but not MD; (3) MD detects but not OD; and (4) Neither OD nor MD detects the object. Of these four cases, only (c) is filtered to be passed to the next process. Cases (1), (2), and (4) are discarded and no longer used. In one frame, there may be several objects that fall into categories (c) and (b) at once. Such a frame will still be forwarded to the next process because it indicates that there is at least one FN object in the frame. These collected FN frames are then labeled using an interactive labeling process before being included in the video stream database.

3.2. Interactive Labeling

Human annotators are typically tasked with localizing objects by drawing bounding boxes and assigning class labels to those boxes. Interactive labeling aims to automate the task of assigning class labels by using an image classifier model. The human annotator's only task is to provide a bounding box in the form of two points: the top-left and bottom-right corners. The interactive labeling speeds up the annotation process and reduce human effort to assign class labels when multiple objects of interest appeared in one frame. Figure 3 illustrates the interactive labeling process. We design the process so that the human annotator is the first to annotate the object's location. This is because *OF* bounding boxes usually do not tightly capture the object due to its movement, making it slower to revise the box than to create a new one. For revising, the human annotator will need to perform four tasks: click the top-left corner of the bounding box, slide it to the correct location, click the bottom-right corner of the bounding box, and slide it to the correct location. Creating a new bounding box will only require two tasks: click the top-left corner of the bounding box and slide it to the bottom-right corner.

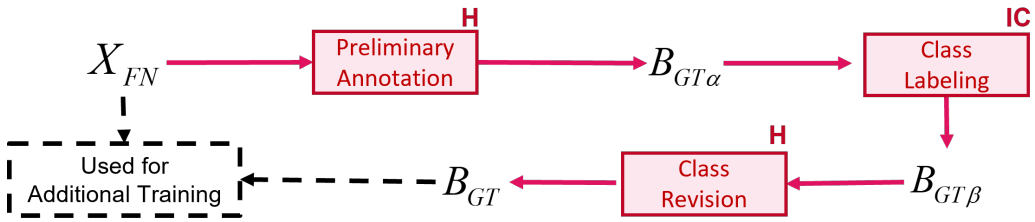

Figure 3. Interactive labeling process flow. Preliminary annotation and class revision is conducted by human (H) while class labeling is conducted by image classifier (IC), a ResNet18 model.

The first interaction with humans is called preliminary annotation. The output is the bounding box of each object of interest in the X_{FN} without a label.

Then, X_{FN} is cropped according to the bounding box $B_{GT\alpha}$, producing images with a single object inside. These images are passed to the image classifier to determine the class of those objects, producing a bounding box with a preliminary class label $B_{GT\beta}$. The image classifier used is a multi-class classifier with a ResNet18 [24] architecture, tuned using a cropped image based on annotations in the VS database. The image classifier is also fine-tuned in each iteration, producing IC-x, where x denotes the period number discussed in detail in Section 3.3. The human annotator checks the class label of $B_{GT\beta}$ and revises it if necessary to finalize the newly annotated data used for additional training, which is X_{FN} and its ground truth label B_{GT}.

3.3. Additional Training

Additional training (AT) refers to the process of adding well-annotated frames as additional data for iterative fine-tuning of the model, as the training process is repeated multiple times with an increasing amount of data until the desired level of accuracy is achieved. Figure 4 illustrates how the amount of data increases with each training cycle. Initially, the base model is created by training the object detector with the base dataset (DB^0), resulting in the base model. DB^0 is a collection of images that have been manually annotated in previous versions of VS, with the exception of the target VS. The database already includes examples of the objects of interest, allowing the OD model's total output to be customized to the number of classes of interest. This process occurs during the D(0) period.

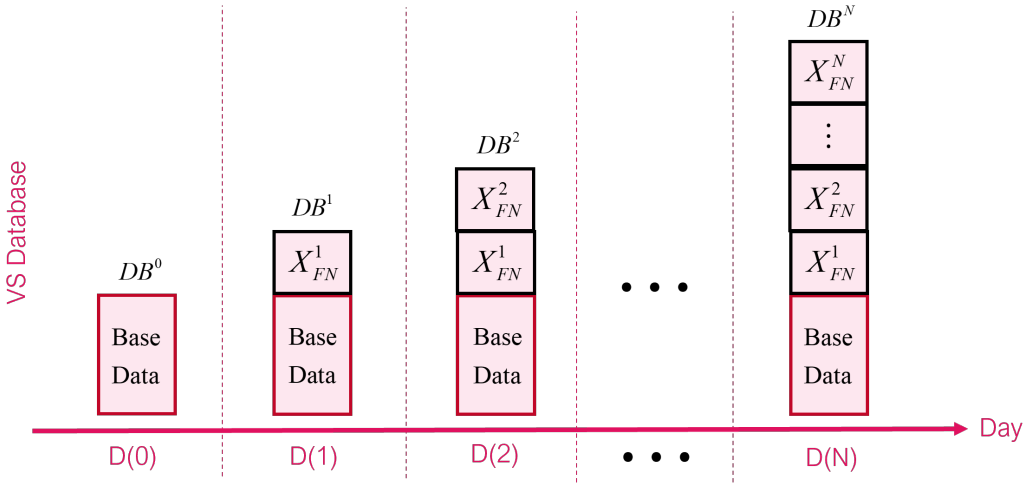

Figure 4. Illustration of how the dataset is updated at each iteration.

AT is carried out during the D(1) period. The length of the day intervals between D(1) and D(0) may vary depending on the frequency of the object's appearance in the VS. If the object appears frequently, then the interval may be one day, but if the object rarely appears, the interval may be longer than one day. During the time interval between D(0) and D(1),

the trained OD model is used for inference on the target VS along with OF in the key-frame filtering process. The results of the data obtained and annotated by the interactive labeling process are then combined with DB^0 to create a new VS database, DB^1.

$$DB^1 = DB^0 + X_{FN}^1 \tag{5}$$

In the D(1) period, we hypothesize that a large number of data will be successfully added because the model has never seen new data in the target VS, thus generating many false negatives. The new VS database is then split into training, validation, and testing sets. Using the new data, the object detector and image classifier are fine-tuned again, resulting in the AT-1 and IC-1 models, respectively. These models are then used to perform inference on the target VS at an interval of D(1) to D(2). This process continues, where the base data for the period from D($k-1$) to D(k) follows the same steps.

$$DB^k = DB^{k-1} + X_{FN}^k \tag{6}$$

That is how the additional training process is repeated until D(N) with final database as DB^N. The AT-N model has reached the desired accuracy target.

4. Experiment Results

4.1. Datasets

The data used to test the effectiveness of the key-frame extraction are from video surveillance in two different locations during the daytime. Both locations have the same task of detecting seven types of objects of interest: (1) tractor, (2) human, (3) bike, (4) truck, (5) loader car, (6) car, and (7) delivery car. The first location has 740 annotated images, with 540 for training, 100 for validation, and 100 for testing. Each image is an RGB image with a size of 1920 × 1080. The seven objects of interest were introduced in 540 training images, which were then used as base dataset (DB^0). Table 1 shows the distribution of each object's occurrence in each set. The other surveillance video is intended as a target VS simulation. The video consists of frames with a rate of 15 FPS with a resolution of 1920 × 1080. Data were recorded for 12 days, from 6 AM to 5 PM. Due to network instability, we only managed to get continuous data records for a minimum duration of 5 s during the day, with a total duration, as shown in Table 2. There can be multiple occurrences of objects of interest in a scene.

Table 1. Distribution of object occurrences (number of instance) in base data.

No	Class	Train	Val	Test
1	Tractor	38	62	11
2	Human	307	9	251
3	Bike	33	9	19
4	Truck	20	3	4
5	Loader Car	125	18	266
6	Car	106	22	52
7	Delivery Car	76	28	263
Total		712	189	862

4.2. Experiment Setup

The experiment was divided into three stages to test the two methods: filtering, annotating, and training. In the manual method, human annotators filter the frames with high variability from the video recordings. The human annotators filter videos taken for three days, which corresponds to D(0) to D(1). In parallel, the key-frame extraction method performs key-frame filtering on the same video. Before filtering, the base model, which is a YOLOv5 model, is trained using the base dataset. The human effort required for the first method is recorded to later compare the efficiency of the key-frame filtering module. After filtering, the data obtained using each method is annotated. The data obtained via

the first method is manually annotated by giving bounding boxes and labels to each object that appears in the frame, using the Computer Vision Annotation Tools (CVAT). The data obtained via the key-frame extraction method is annotated using the interactive labeling method. Before annotating, the image classifier is trained using the base data. The time taken by humans to complete annotations in both methods is recorded.

Table 2. Target vs. Record Duration.

Day	1	2	3	4	5	6	7	8	9	10	11	12	Total
Total Video Duration (h)	8.3	11.1	5.3	5.7	11.7	12.3	11.2	11.8	11.4	8.7	8.6	12.1	118.1
Number of videos	37	41	23	12	26	23	24	24	28	49	17	23	327

The annotated data are then randomly split into training, validation, and testing sets with a percentage ratio of 70:15:15 for manually gathered data and 75:25:0 for key-frame extraction. This ratio is based on the number of images and object occurrences of each class. The data obtained using key-frame extraction is not split into the test set because the accuracy of the model with key-frame extraction data will be tested using the testing set on the manually annotated data, which has more diverse data as it includes both FN and FP data. The split data are then merged with duplicates from the base data to create two databases: VSD-M(1) for the base data merged with annotations obtained via the manual method, and VSD-K(1) for the base data merged with annotations obtained using the key-frame extraction method. The base model is then fine-tuned using the VSD-M and VSD-K data to produce the AT-M-1 and AT-K-1 models, respectively, which are tested using the test set on VSD-M(1). The fine-tuning process is the same for both models, running the training for 100 epochs with early stop using the same loss function and optimization method used by the YOLOv5 author. The results obtained via the AT-M-1 and AT-K-1 models are then compared by calculating the mean average precision (*mAP)* with an IOU threshold of 0.5. This metric is widely used to test the accuracy of the object detection model [4,8,20] using the following:

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \tag{7}$$

where, n represents the number of classes. We set n to seven in the experiment. The filtering, annotating, and additional training processes are then repeated three times, carried out on the 20th–22nd, 23rd–25th, and 26th–28th of October. The accuracy improvement is recorded for each period, along with the human effort required to complete the cycle.

4.3. Results and Analysis

Table 3 shows the results of our experiments as described in the scenario in Section 4.2. There are four D(x) periods, and each key-frame extraction cycle required a total of 5.2 times less human effort or reduced manual effort by 81% (Figure 5). We calculate this duration by focusing only on the parts that cannot be replaced by humans, such as filtering important frames from surveillance video and annotating selected frames. Processes that can be automated, such as organizing files, starting the model training, and testing the model, are ignored.

Table 3. Total human effort used using manual and the key-frame extraction method on each cycle.

D(x)	Manual			Key-Frame Extraction (m)
	Filtering (m)	Annotating (m)	Total (m)	
1	122.4	80.5	202.9	63.8
2	133.9	105.7	239.6	36.8
3	179.6	133.1	312.7	50.8
4	126.7	89.7	216.4	34.1

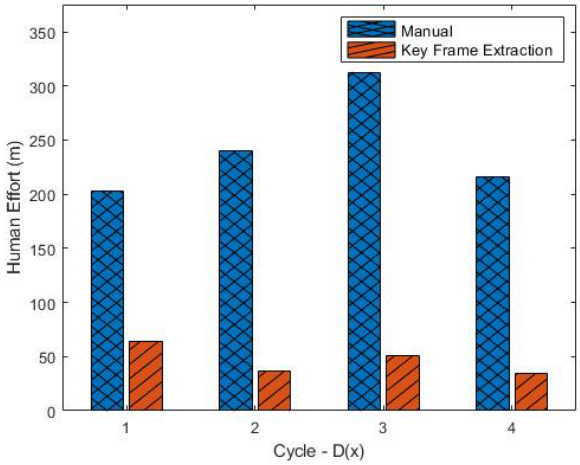

Figure 5. Summary of human effort needed for manual and key-frame extraction methods.

For the manual method, the filtering and annotating processes take time. Our experiment results show that the filtering process takes longer than annotating. The filtering process is carried out at a video speed of 25×, and the goal is to crop the part of the video that has at least one object of interest. The time it takes depends on the duration of the video and how many objects pass through the scene during that period. This can be seen in D(1) and D(4), which require longer filtering than D(2) and D(3), because the video duration during those periods is longer. We found that the average time required for manual filtering was 4.8 min per hour of video recording. On the other hand, the annotating process is also highly dependent on how many frames were successfully filtered during that period. The more frames that need to be annotated, the longer the annotation process will be. This does not depend on the duration of the video to be filtered, but only on how many objects pass through the video surveillance. We found that the average time required for manual annotating was 21.6 s per frame.

The key-frame extraction method requires only the annotating process, as the filtering process is carried out automatically by the key-frame filtering module, and therefore does not require human effort. Our experimental results show that the average time needed for interactive labeling is 17.3 s per frame, which means a reduction of 20% in the annotation process time. The details of this time calculation will be discussed in Section 5.3.

Figure 6 shows the accuracy of the YOLOv5 nano model trained with VSD-M(x) and VSD-K(x) data resulting in the AT-M-x and AT-K-x models, respectively. We measured the accuracy based on the mAP metric with a threshold of 0.5. Our experimental results show that the accuracy obtained between AT-M and AT-K is quite similar. The comparison between the accuracy of AT-M-x with the corresponding AT-K-x shows only a 0.2–0.3% difference. This indicates that the key-frame extraction method does not lead to a decrease in accuracy. The accuracy values of the two models are also significantly higher than the base model that was not trained with data from the target VS. This highlights the importance of introducing new data in the use of supervised learning-based models, such as YOLOv5.

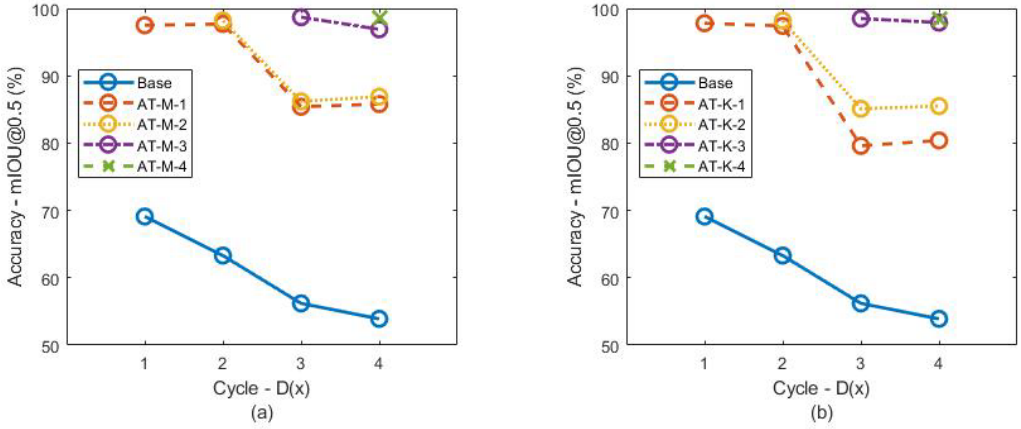

Figure 6. Accuracy of the model trained by (**a**) manual and (**b**) key-frame extraction method on the test set calculated by mIOU metric with a threshold of 0.5.

Our experiments also show that there is a decreasing trend in the accuracy of the model as the amount of data increases. As seen, both AT-M-1 and AT-K-1 have lower accuracy in the fourth cycle. This is because the amount and variability of the data are getting higher so that the accuracy drops again. This indicates that additional learning is necessary until we achieve a high accuracy. We also observed a change in trend between D(3) and D(4), where the accuracy of each model AT-M-x and AT-K-x from 1–2 showed an increase in accuracy. This is due to the increase in data in D(4), which has many similarities with the data taken in D(1) and D(2), leading to an increase in the number of similar test cases. Consequently, both models report a slight increase in accuracy.

5. Ablation Study

5.1. Key-Frame Filtering Effect

We also recorded the number of frames that were collected and finally annotated for use in the training process. The amount of data collected is not the same as the amount annotated because there are consecutive frames that are still too similar to be used in the training process. Cases like these are very common in frames that are collected using the manual method (Figure 7). This is because in the manual frame filtering process, the method used is to crop videos with objects of interest moving in them and then take frame pieces every 1 s. For objects of interest, such as motorcycles and various types of cars, 1 s is long enough to provide many different conditions, whereas for objects, such as humans, which tend to have slower movement, the frames will still be similar. Experimental results show that for the manual case, the ratio of collected and annotated data is close to 10:1.

The data collected via key-frame filtering is much smaller. The ratio between manually collected frames and key-frame filtering is close to 13:2. The small amount of data that needs to be annotated also contributes to reducing human effort in the overall key-frame extraction process compared to manual filtering. This shows the effectiveness of key-frame filtering in capturing only important frames without involving human intervention. However, not all captured frames are suitable for annotation. Some frames may contain movement of objects that are not part of the object of interest, such as birds, dogs, and the movement of tree leaves. This is expected since the key-frame filtering module is designed to capture all movements in the video surveillance scene. From the experimental results, the ratio between frames collected and annotated with our key-frame filtering method is 11:4.

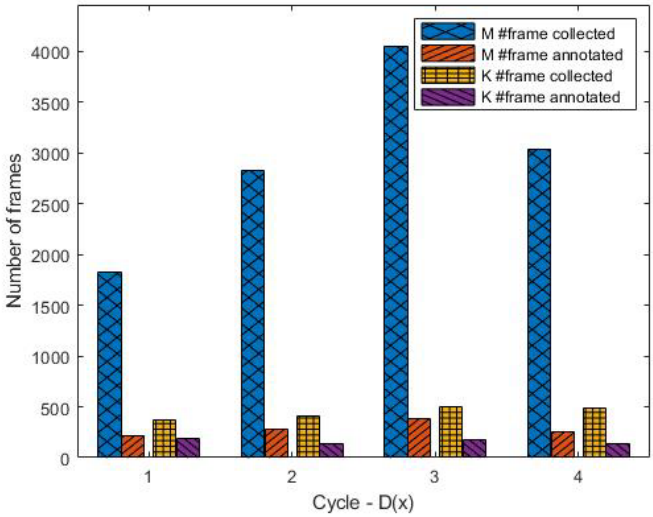

Figure 7. Number of frames collected and annotated via manual and key-frame extraction method.

5.2. Classifier Evolution

We tested the accuracy of the interactive labeling tool that we used in every cycle. The accuracy is calculated using a simple metric, which is the number of correct predictions without correction from the human annotator compared to all predictions. Figure 8 shows the accuracy of the interactive labeling tool. As the amount of data increases in the next cycle, there is a downward trend in the accuracy of the IC. Similar to YOLOv5, the IC-1, which is not trained using data from the target VS, has a low accuracy in this case, below 60% in each cycle. However, on IC-2, the accuracy increases to 95%, although it drops again when used in D(3) and D(4). This shows that ResNet18 as an IC also requires additional training, similar to YOLOv5. However, this additional training process does not require additional human effort, as all processes, such as frame extraction for additional ResNet18 training, can be conducted automatically using a scripted command.

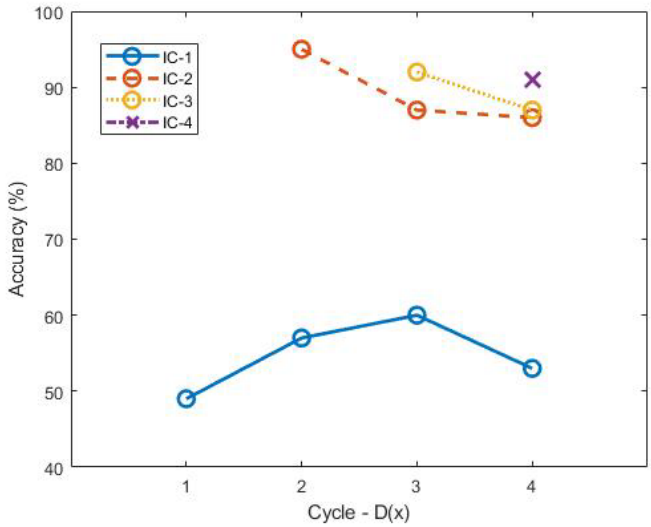

Figure 8. Accuracy of the image classifier during the interactive labeling.

5.3. Interactive Labeling Breakdown

We provide details on the distribution of human effort in the interactive labeling process. In Section 3.2, interactive labeling is divided into two phases: preliminary and revision. Figure 9 shows the amount of human effort spent on each phase in each cycle. Experiments show that the preliminary phase requires more effort, on average nearly three times the time required for the revision phase. This is expected because in the preliminary

process, the human annotator requires more interaction since they have to move the cursor to each position of objects that are not close together. In the case of revision, the position of the changed class is not far away since there are only seven selected classes. We hypothesize that this time distribution may change depending on the number of classes of the object of interest.

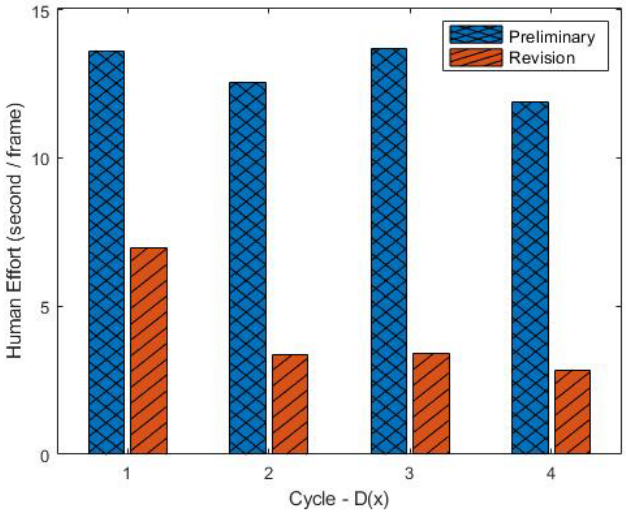

Figure 9. The amount of human effort spent on each phase in each cycle.

Our experiments also show that there is a downward trend in the time required for revision. In D(1), the ratio between the preliminary and revision time is 2:1. Then, it decreased significantly in D(2) to 3.7:1 and decreased gradually to 4.2:1 in D(4). This is because the new IC is trained using target VS data in D(2). As the amount and variability of the data used to train ICs increases, the accuracy also gets better so that the time required for revision tends to decrease. This result is related to the evolution of the image classifier.

5.4. Training Breakdown

Using the key-frame extraction method reduces the training duration of YOLOv5. The training duration consistently had an upward trend in each cycle, which is due to the increasing amount of data in each cycle (Figure 10). However, in terms of the duration comparison between manual and key-frame extraction, there is a significant gap. At D(1), the key-frame extraction takes only 11.5% less time than manual, but this gap further increases linearly to 29.1% in D(4).

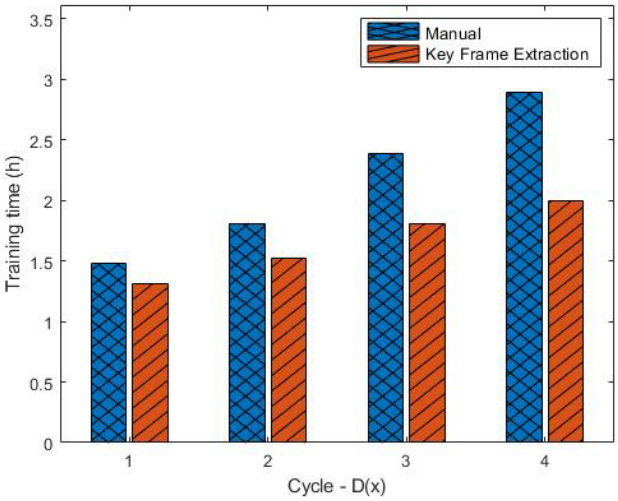

Figure 10. Training duration of the YOLOv5 models.

The training duration is reduced on average by 21.9% since the amount of data filtered and successfully annotated in the manual process is the same, while in key-frame extraction, this is reduced because it uses key-frame filtering. The IoU comparison module in key-frame extraction discards many frames as the YOLOv5 becomes better. The small number of frames makes the additional training process faster because it takes less time to complete one epoch. This reduced training duration is beneficial for scaling object detection systems since fewer resources are needed to meet the demands.

6. Conclusions

This paper presents a framework for reducing human effort in building object detection models for video surveillance, making them more scalable. The framework consists of two components: key-frame filtering and interactive labeling. These components help prepare new data for use in the fine-tuning process with less human effort. The fine-tuning process is carried out periodically. The key-frame filtering component replaces the need for manual frame filtering by using the static background and moving objects of interest present in video surveillance. The interactive labeling component reduces the human effort required for annotating object detection data by using an image classifier to determine the class label, while the human annotator is tasked only with providing bounding boxes.

We tested the effectiveness of key-frame extraction by conducting experiments on a dataset consisting of video surveillance data taken from two different locations. We compared the human effort required to train the YOLOv5 model to perform object detection at the second location, given that the model has been trained from the data at the first location. Experiment results showed that key-frame extraction reduced human effort by 81% compared to the manual method. The accuracy of mAP with a threshold of 0.5 from the two models also showed a slight difference of 0.2–0.3%. We also conducted ablation studies to see the effect of the key-frame extraction. We found that key-frame filtering succeeded in selecting fewer frames, which is 2:13 compared with the data filtered manually, thereby reducing human effort during annotation. The time required to train the model using the key-frame extraction method is 25% less than the manual method.

Author Contributions: Conceptualization, S.G.K.; Methodology, H.R.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Faculty Research Fund of Sejong University (2021).

Data Availability Statement: The data are not publicly available due to a non-disclosure agreement.

Conflicts of Interest: No potential conflicts of interest relevant to this article reported.

References

1. Mandal, V.; Mussah, A.R.; Jin, P.; Adu-Gyamfi, Y. Artificial intelligence-enabled traffic monitoring system. *Sustainability* **2020**, *12*, 9177. [CrossRef]
2. Golcarenarenji, G.; Martinez-Alpiste, I.; Wang, Q.; Alcaraz-Calero, J.M. Machine-learning-based top-view safety monitoring of ground workforce on complex industrial sites. *Neural Comput. Appl.* **2022**, *34*, 4207–4220. [CrossRef]
3. Lee, M. Iot livestock estrus monitoring system based on machine learning. *Asia-Pac. J. Converg. Res. Interchange* **2018**, *4*, 119–128. [CrossRef]
4. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
5. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
6. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
7. Zhang, D.; Han, J.; Cheng, G.; Yang, M.-H. Weakly supervised object localization and detection: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 5866–5885. [CrossRef] [PubMed]
8. Yao, A.; Gall, J.; Leistner, C.; Van Gool, L. Interactive object detection. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3242–3249.
9. Koohbanani, N.A.; Jahanifar, M.; Tajadin, N.Z.; Rajpoot, N. NuClick: A deep learning framework for interactive segmentation of microscopic images. *Med. Image Anal.* **2020**, *65*, 101771. [CrossRef] [PubMed]

10. Farnebäck, G. Two-frame motion estimation based on polynomial expansion. In Proceedings of the Scandinavian Conference on Image Analysis, Halmstad, Sweden, 29 June–2 July 2003; pp. 363–370.
11. Wang, G.; Li, W.; Zuluaga, M.A.; Pratt, R.; Patel, P.A.; Aertsen, M.; Doel, T.; David, A.L.; Deprest, J.; Ourselin, S.; et al. Interactive medical image segmentation using deep learning with image-specific fine tuning. *IEEE Trans. Med. Imaging* **2018**, *37*, 1562–1573. [CrossRef] [PubMed]
12. Fukushima, K.; Miyake, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and Cooperation in Neural Nets*; Springer: Berlin/Heidelberg, Germany, 1982; pp. 267–285.
13. Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698. [CrossRef]
14. LaBonte, T.; Song, Y.; Wang, X.; Vineet, V.; Joshi, N. Scaling novel object detection with weakly supervised detection transformers. *arXiv* **2022**, arXiv:2207.05205.
15. Bilen, H.; Vedaldi, A. Weakly supervised deep detection networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2846–2854.
16. Wang, C.; Ren, W.; Huang, K.; Tan, T. Weakly supervised object localization with latent category learning. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 431–445.
17. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
18. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv* **2022**, arXiv:2209.02976.
19. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
20. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
21. Sorokin, A.; Forsyth, D. Utility data annotation with amazon mechanical turk. In Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, San Francisco, CA, USA, 13–18 June 2008; pp. 1–8.
22. Welinder, P.; Perona, P. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, San Francisco, CA, USA, 13–18 June 2010; pp. 25–32.
23. Suzuki, S.; be, K. Topological structural analysis of digitized binary images by border following. *Comput. Vision Graph. Image Process.* **1985**, *30*, 32–46. [CrossRef]
24. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.