CL-Shield: A Continuous Learning System for Protecting User Privacy

Tianyu Li, Hanling Wang, Qing Li, *Senior Member, IEEE*, Yong Jiang, *Member, IEEE*, and Zhenhui Yuan, *Senior Member, IEEE*

***Abstract*—The video analytics system utilizes deep learning models (DNN) to perform inference on the videos captured by cameras. Continuous learning algorithms are used to address the data drift problem in video analytics systems. However, uploading images from deployment environments and processing on the cloud carry the risk of privacy leakage. In this paper, we have designed a system called CL-Shield to protect user's privacy. Firstly, we review the causes of privacy leakage in a continuous learning system and propose the objective of full privacy protection. Secondly, we design an online training mechanism based on a scene library to avoid direct uploading of user's frames to the cloud server. Lastly, we design a fast training set search algorithm based on a novel Ebv-List, which effectively improves the speed of model updates. We collect various real-world scenario data to build our scene library and validate our system on a dataset of over 10 hours. The experiments demonstrate that our privacy-aware continuous learning system achieves an F1-score of over 92% compared to the conventional systems without protecting privacy and has long-term stability in analytic F1-score.**

***Index Terms*—Video analytics, Continuous learning system, Edge computing, Privacy protection**

I. Introduction

The video analytics system [1]–[4] utilizes deep neural networks (DNN) to analyze videos captured by cameras. These systems are widely deployed in a variety of fields such as individual tracking [5], city surveillance [6], action recognition, and others, in both industrial and everyday settings. Lightweight models are often deployed for inference on edge devices to minimize end-to-end latency. However, the limited capacity of lightweight models, due to their smaller number of parameters, restricts their ability to capture features across a broad range of scenes and objects (see Fig.1), which can lead to data drift [7]. The primary approach to resolving data drift involves updating the lightweight models online by conducting training processes on cloud servers. The use of continuous learning mechanisms enhances the accuracy of video analytics tasks. However, state-of-the-art continuous learning approaches [6]–[9] suffer from privacy leaks. Online training of lightweight models requires the collection of video segments or frames from the environment. The transmission of frames containing environmental data to the cloud server and subsequent processing on the server can potentially lead to privacy breaches, as illustrated in Fig. 2, limiting the adoption of continuous learning systems.

Tianyu Li and Hanling Wang are with Pengcheng Laboratory, Shenzhen, Guangdong 518055, China, and also with the Shenzhen International Graduate School, Tsinghua University, Shenzhen, Guangdong 518055, China (e-mail: lity21@mails.tsinghua.edu.cn, hl-wang21@mails.tsinghua.edu.cn).

Qing Li is with Pengcheng Laboratory, Shenzhen, Guangdong 518055, China (e-mail: liq@pcl.ac.cn).

Yong Jiang is with the Shenzhen International Graduate School, Tsinghua University, Shenzhen, Guangdong 518055, China, and also with Pengcheng Laboratory, Shenzhen, Guangdong 518055, China (e-mail: jiangy@sz.tsinghua.edu.cn).

Zhenhui Yuan is with the School of Engineering, University of Warwick, Coventry CV4 7AL, UK (e-mail: zhenhui.yuan@warwick.ac.uk).

Manuscript received xxx. *(Corresponding author: Qing Li.)*

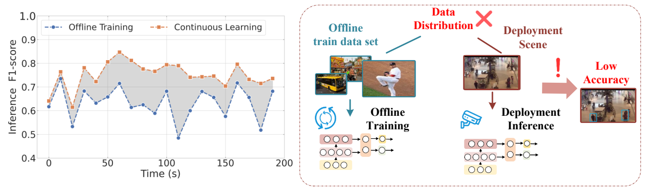


Fig. 1: Data drift. The left subfigure illustrates that data drift will lead to accuracy loss compared to continuous learning on the Mall [10] dataset. The right subfigure demonstrates that data drift occurs due to inconsistency distribution between offline training data and deployment environment.

Privacy protection in continuous learning systems possesses unique characteristics distinct from conventional video analytics systems that do not involve model updates or storage systems. Video analytics inference algorithms [11] only furnish statistical data and are inadequate for real-time training purposes. Users encrypt their images before transferring them to cloud servers to prevent comprehensive access by cloud servers. Although servers temporarily store user images to overcome storage limitations on users' devices, cloud servers cannot access these encrypted images. Solutions proposed by Tian [12] extract motion and location data directly from encoded data streams to prevent leakage; however, the method is tailored for inference tasks and lacks the capability for training through back-propagation, rendering them unsuitable for continuous learning systems.

Challenges: It is essential to solve the privacy leakage problem in continuous learning systems. In our design, we encountered two challenges: (1) **Privacy information is difficult to capture.** Both foreground and background information could be exploited by unauthorized individuals. Current approaches, such as target detection, cannot accurately identify foreground targets and can't eliminate background privacy information. (2) **Online training demands high-quality images.** Continuous learning relies on training with images of the same scenes used for edge model inference, which requires high-quality training

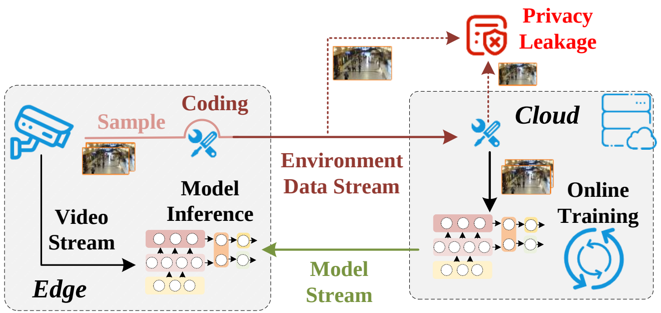


Fig. 2: Privacy leakage in continuous learning systems.

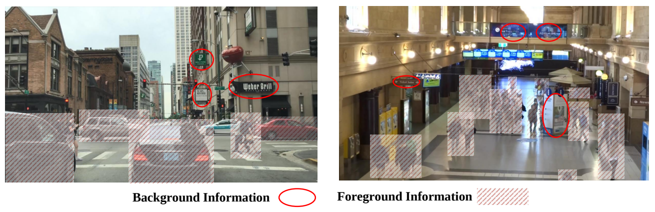


Fig. 3: Private information is present in both background and foreground.

images.

Solution: To address these challenges, we have proposed a new continuous learning system called CL-Shield. First, we define the "privacy protection" issue in continuous learning systems and analyze the causes of privacy leakage. We then propose the "full privacy protection" concept in continuous learning systems, which aims to protect foreground information and background information of the deployment environment while achieving the goal of continuous learning accuracy improvement. The aim of this paper is to set high privacy standards that align with the role of continuous learning systems themselves.

In the system design, we propose the concept of a "scene library" which stores images of various scenes. The system can select images of similar scenes based on the characteristics of the deployment environment for the online training process, which avoids directly collecting images from the deployment environment and effectively protects user privacy. The next step is to find optimal training sets quickly from thousands of images. Finding similar scenes should achieve the following three goals: (1) High matching accuracy, (2) High matching speed, and (3) The ability to quickly update the scene library and search data structure. To address this issue, we have designed a new hash table-based search method that supports real-time expansion of the scene library and rapid mapping searches. Additionally, to further utilize the temporal correlation of videos, we have innovatively designed the Ebv-List (short for "list of excluding bad vectors"), which records previous searches to minimize unnecessary comparisons. The fast search algorithm based on the Ebv-List can save time in finding similar scenes, thus allowing ample time for online training of lightweight models. We want to emphasize that the design of CL-Shield is not oriented towards class-incremental learning scenarios [13]–[16]. Our contributions in this paper can be summarized as follows:

1) We analyzed the privacy leakage problem in continuous learning systems and set the objective of full privacy protection in the continuous learning system.
2) This paper proposes an algorithm based on a "scene library", which provides scene images similar to the deployment environment for the online training process of continuous learning systems, thereby avoiding direct upload of user images to cloud servers.
3) This paper proposes an accelerated search strategy based on Ebv-List, which can quickly find the appropriate scene in the "scene library" for online training processes.

We collected real-world data to verify the performance of the system. Experimental results show that our system is the only one among all baseline algorithms that is capable of providing "full privacy protection" and enhancing accuracy. Our privacy-protected continuous learning system can achieve an F1-score of 92.1% - 99.77% of existing continuous learning systems without privacy protect. We conducted experiments on continuous videos lasting 10 hours and demonstrated that our system is not affected by forgetting, which leads to a decrease in F1-score.

II. Background and Motivation

A. Analysis of Privacy Leakage

The generalized continuous learning system consists of two deployment machines (edge and cloud) and two transmission links. The two transmission links are the uplink transmission link (data stream for transmitting deployment environment information) and the downlink transmission link (data stream for transmitting updated model). The management of the data on these four parts is summarized in Table I. The existence form of the above data in these two parts (uplink and cloud) can directly reflect the real information of the deployment environment, and the risk of user privacy leakage is large. Therefore, it is necessary to design a new continuous learning system to first reduce the leakage of environmental information, and then improve the user's trust in the use of video analytics system.

Unlike prior work such as Privid [11] on video analytics systems, which solely concentrate on model inference tasks, we introduce the following two objectives in designing the continuous learning systems:

(1) Objective 1 - accuracy improvement: We use V to denote the collection of video segments, and use f to denote the model trained offline. The continuous learning system has two types of accuracy: one is the inference accuracy of the lightweight model on video segments $v \in V$ without online training $A(v, f)$, the other is the inference accuracy $A(v, f_{on})$ after online learning. Adding some privacy-preserving measures of a continuous learning system yields an accuracy on $A(v, f_{p,on})$. The theoretical accuracy range that a privacy-preserving continuous learning system should achieve is shown as Equation (1), and the system design should make

TABLE I: Privacy risk analysis of each part of continuous learning system

Name of components	Managers	Forms of data	Probability of privacy leakage	Types of leakage	User trust
Natural environment	User	Real environment	Low	None	High
Sensing device	User	Video or image	Low	Storage leakage	High
Edge device	User	Video or image	Low	Storage leakage	High
Upload link	Network operator	Image encoding stream	High	Transmission leakage	Low
Download link	Network operator	Model encoding stream	Low	Transmission leakage	High
Cloud server	Service provider	Image and model data	High	Storage leakage	Low

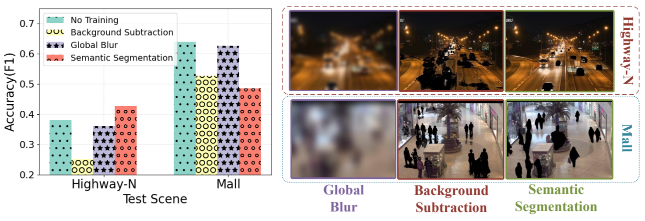


Fig. 4: Limitations of traditional methods. The sub-figure on the left shows that using traditional privacy-preserving methods results in a loss of F1-score in continuous learning systems. The sub-figure on the right shows images processed with traditional privacy-preserving methods.

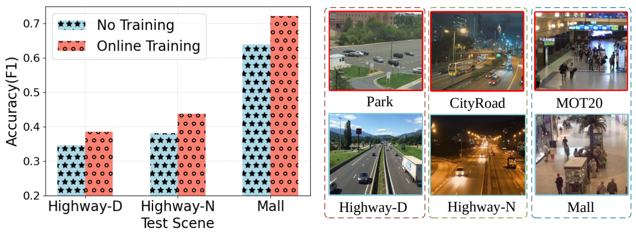


Fig. 5: Improved F1-score through training on similar scenes. The two sets of data within the dashed vertical box are taken from similar but distinct scenarios. The data outlined in red (top) is used for online training, while the data outlined in blue (bottom) is used for testing.

the system accuracy as close as possible to the upper limit $A(v, f_{on})$.

$$A(v, f_{p,on}) \geq A(v, f), \quad A(v, f_{p,on}) \leq A(v, f_{on}) \qquad (1)$$

(2) Objective 2 - full privacy protection: As shown in Fig. 3, the video collected by the camera contains a great deal of privacy information, which can be divided into two categories: (1) Private information contained in the foreground, such as vehicle color, type of objects, license plate, pedestrian pose, the face of a person, and the person in the car. (2) private information about the context (also called inferential privacy [17]), such as traffic signs, store signs, building locations, and so on. Private information in the foreground has been studied widely, but the background still reveals personal information such as whereabouts. The privacy protection method explored in this paper not only protects the information security in the foreground, but also protects the information security in the background. In other words, we will design a new system to hide and protect most of the environmental information that can be directly observed and understood by the human eye before it is uploaded to the cloud server in the process of uploading and online training.

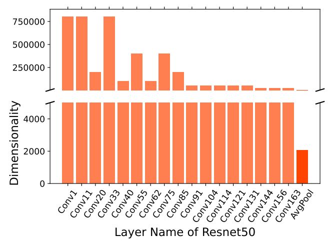
B. Limitations of Traditional Measures

In order to deal with the problem of user privacy leakage brought by the uploading data, the traditional approach is to add noise to the image. There are two typical ways to add noise. (1) Protect the global information by blurring the image with a Gaussian kernel. (2) Protect the foreground by erasing it, such as using semantic segmentation (DeeplabV3 [18]) to identify the foreground or using the algorithm [19] based on background modeling.

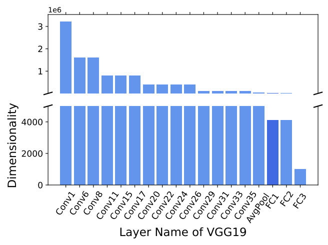
Fig. 4 (left) shows the F1-score of the continuous learning system after the frames are processed by these methods, and Fig. 4 (right) shows the images after these processes. Images processed with semantic segmentation (DeeplabV3-Resnet101) might lead to F1-score improvement, but objects in the foreground are not fully hided, resulting in privacy leakage (violation of objective 2). Although the background subtraction algorithm can identify most of the moving objects and play a certain role in privacy protection, the F1-score after continuous learning is damaged (violation of objective 1 and objective 2). The algorithm based on global blur hides both foreground private information and background private information at the same time, but the F1-score of the continuous learning is undermined (violating objective 1). This experiment also proves that the traditional privacy protection methods violate the objectives proposed in this paper and cannot be directly transferred to the design of the continuous learning system.

C. Benefit of Using Similar Images

Ekya [7], RECL [9], and others have collected frames from the system deployment environment to train lightweight models. However, our experiments have shown that using frames from similar scenes for online training can also improve the detection accuracy of lightweight models. Here, we fine-tune the Efficientdet-D0 model using the Park [20], CityRoad [21], and Mot20 [22] datasets to obtain three models, referred to as online training, while the model before online training is referred to as no training. Then, we conduct inference on frames from the Highway-D [23], Highway-N [24], and Mall [10] videos using these three models, resulting in the outcomes shown in Fig. 5. The Highway-D and Park frames are both from daytime data, with the same target objects (vehicles) in the images. Both Highway-N and CityRoad frames are from nighttime data, while the Mall and Mot20 frames are



(a) ResNet50



(b) VGG19

Fig. 6: Information compression through deep neural network layers.

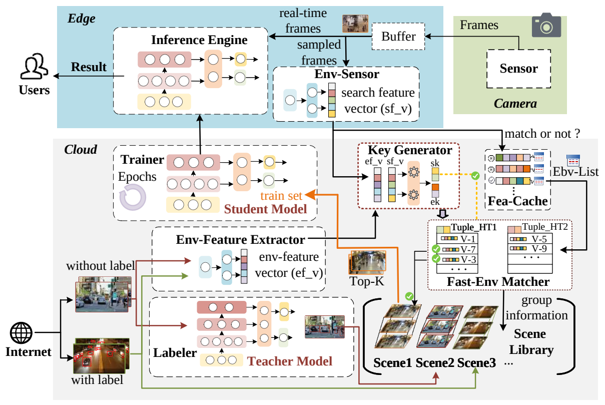


Fig. 7: System overview.

from indoor crowded scenes. Although these three sets of data are from different deployed environments, they exhibit certain similarities in terms of foreground objects and backgrounds. As shown on the left side of Fig. 5, the online training mode using frames from similar environments for continuous learning can bring about an F1-score improvement of 11.3% to 14.7% compared to no training.

D. Privacy information transformation

Illustrated in Fig. 6a and 6b, by visualizing the output sizes of intermediate layers of ResNet50 and VGG19, we can draw the following conclusions: (1) Deeper layers compress high-dimensional raw images (e.g., two million bytes of data from a 1080p image) into low-dimensional features (e.g., the output data of the global average pooling layer in ResNet50 is only 2048 dimensions, and the output of the FC1 layer in VGG19 is only 4096 dimensions). These deep features have already lost the visual information of the original image and can be utilized to transform privacy information embedded in images. (2) Information extracted by the shallow layers of DNN contains a substantial amount of data and carries easily discernible visual information, making it vulnerable to privacy leaks during transmission and on cloud servers.

III. System Model

A. Overall System Design

Based on the motivation above, we propose a system named CL-Shield, which is a privacy-preserving continuous learning system. The structure of the system is shown in Fig. 7. CL-Shield is deployed on the edge and the cloud. The edge has a lightweight object detection model (Lw-Model). The cloud is mainly responsible for the online training process of the Lw-Model. Unlike previous work [7]–[9], the labels for the training dataset in the online learning process of CL-Shield come from both the teacher model and manual annotations (public dataset). As Fig. 2 shows, CL-Shield has two data streams between the edge and the cloud: environment data stream and model data stream. CL-Shield's environment data flow is different from the way previous work uploads images, and the whole process is more secure. We divide the whole working process of the video analytics system into several online learning windows with equal time, abbreviated as win_{ol}. Like AMS [8], the length of win_{ol} is limited to 10s in this paper.

Edge: Edge devices capture video from the camera and decode it into video frames that are stored in the buffer. The real-time Inference Engine (Lw-Model deployed) is used for real-time video inference, and the analysis results are fed back to users or upper applications in time. The Env-Sensor (deployed lightweight feature extraction network) inputs sampled frames, and processes the searching feature vector sf_v. Then the edge will transmit sf_v to the cloud for similar scene lookup. At the beginning of each continuous learning window win_{ol}, the Env-Sensor collects one frame of the video. As a result of this method, privacy is preserved since we do not upload frames from the deployment environment, but rather the feature vector generated by the search.

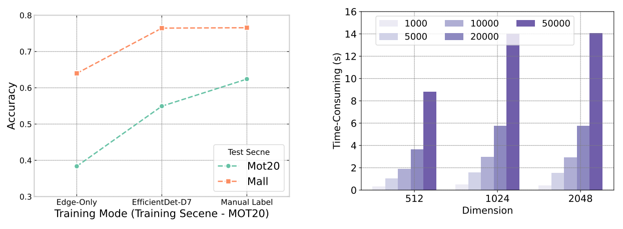
Cloud: A Scene Library is dynamically maintained on the cloud, which stores the image and its labels in various scenes (the definition of the scene will be explained later). The Trainer obtains (picture, label) information from the scene library for online training of the Lw-Model, and sends the new model weights to the edge. Like Ekya [7] and RECL [9], each win_{ol} updates all the parameters of the Lw-Model.

An important aspect of online training is the fast matching of similar scenes, which is performed by the Fast-Env Matcher. The overall idea of CL-Shield can be described as to find scene frames from the scene library that are similar to the deployment environment, and then use these frames for the online training process on the cloud server.

B. Establishment of Scene Library

In this section, we introduce the scene library designed for the continuous learning system.

Deployment environment descriptor: Existing continuous learning systems [7]–[9] send representative images from the edge to the cloud for online training. In order to protect user privacy, the goal of our system design is to avoid sending visually comprehensible environmental information to the cloud server as much as possible. To inform the cloud about the current deployment environment, at time t, where $t \in T$, we



(a) Manual label

(b) Linear search time

Fig. 8: Design choices.

use the Env-Sensor m_f to obtain the features (referred to as the environment descriptor) of the deployment environment v and send it to the cloud server via the upload link. The process of obtaining the environment descriptor is recorded as symbol $E_{fea}(\cdot)$. Throughout the entire testing period T, the system maintains a sampling rate of r_s FPS (frame per second), and the work process of the Env-Sensor can be expressed as in Equation (2).

$$sf\_v_t = E_{fea}(v, r_s, m_f), v \in V \tag{2}$$

Definition 1 - scene: A collection of frames $\{p_i | i \in I_q, q \in Q\}$ captured by the same camera at the same location within a short period of time (for example, within a few minutes) constitutes a scene sc_q, where Q represents the index sequence of scenes, and I_q is the index number of the frames collected in scene sc_q.

Scene library: In order to protect users' privacy, this paper establishes a train set buffer called scene library that integrates various scenes. In a scene library, the larger the variety of images, the easier it is for the system to locate scenes that are similar to the environment where the camera will operate. Therefore, we have collected as many images as possible, both online and offline. It should be noted here that the scene library contains scenes that are different from those captured by sensors. A description of the data source for the scene library can be found at IV-B. Admittedly, the improvement in the accuracy of continuous learning designed in this paper is limited by the quality of the scene library.

Image representation vector: As in Equation (3), Env-Feature Extractor use the m_f to extract the features (ef_v_i) of the image p_i and add it to the scene library. The extracting process is recorded as symbol $E_{rep}(\cdot)$.

$$ef\_v_i = E_{rep}(p_i, m_f) \tag{3}$$

AMS [8] and RECL [9] deploy the "Golden Model" on the cloud to provide the label $label\_gm$. The inference process of "Golden Model" (represented as symbol E_{label}) can be described as Equation (4):

$$label\_gm_i = E_{label}(p_i, m_g) \tag{4}$$

where m_g is the "Golden Model" used to provide the labels.

As in Fig. 8a, we train the object detector EfficientDet-D0 using the data from MOT20 [22] scene, and then verify it on MOT20 and Mall [10] respectively. Three modes are selected in the experiment: (1) Edge-Only without any fine-tuning measures, (2) the labels of the second training set are from EfficienDet-D7, and (3) the labels of the third mode are provided by humans. Experimental results show that higher F1-score can be achieved by using artificial labels. In addition, due to the excessive reliance on the reasoning ability of the "Golden Model" in the traditional continuous learning system, there are also problems such as excessive demand for computing resources and long delay in obtaining labels in some scenarios. If the manual label $label\_man$ is used for labeling, it will consume too much manpower and have a longer delay to obtain the label, making it difficult to deploy in an automated system. In order to solve the above problem, we propose a hybrid label generation method to provide the required labels for the online training process. Next, we illustrate the specific operation steps of the proposed method.

Mixed label generator: There are two types of images that are currently available through public sources: (1) Data (D_α) that only contains image information, such as videos from YouTube; (2) Data (D_β) contains image and its human labels ($label_{man}$), such as the COCO dataset [25]. In order to expand the scenarios in our scene library, we design a mixed label generator which can be expressed as Equation (5). CL-Shield uses it to provide labels for training lightweight model. By using two data sources, CL-Shield expands the types of scenes in the scene library, thereby reducing the possibility of bias between online training data and deployment scenarios.

$$label_i = \begin{cases} label\_gm_i & \text{if } p_i \in D_\alpha \\ label\_man_i & \text{if } p_i \in D_\beta \end{cases} \tag{5}$$

Above, we have explained how to obtain the image p_i, representation vector ef_v_i, and $label_i$. Then we combine them into a tuple $scene\_tuple_i(p_i, ef\_v_i, label_i)$ and add it into the scene library. CL-Shield retains the external interface, using the "Golden Model" to infer the newly added frame of the scene or directly using artificial label information. The scene library can be expanded online, without affecting other online processes in cloud.

Online learning process: The online learning process is carried out within the Trainer on the cloud, which is responsible for the online training of the Lw-Model. At the beginning of win_{ol}, the system selects (the selection method is described in Section III-C) the Top-K images with the highest matching scores from the scene library as the training set s_w. Subsequently, the Trainer uses the training set s_w to train the student model, resulting in new model weights. Once the weights have been updated, they are sent to the edge. Similar to AMS [8], in order to avoid disrupting the real-time inference process of the Lw-Model on the edge, a copy of the model is maintained on the edge. The updated model weights are first loaded into the copy, and then the model copy is swapped with the execution model in the real-time inference.

Next, we will discuss how to quickly and accurately search for similar scene images in the scene library for online training.

C. Organization of the Scene Library

We need to locate images in the scene library that are similar to the deployment environment for the online training process. There are three key points in the similar scene search process: (1) How to determine whether two scenes are similar? (2) How to efficiently structure the images in the scene library and support real-time insertion of new data? (3) How to quickly search for similar images in the scene library?

Definition 2 - similarity of scene: We use cosine similarity to calculate the matching degree between the deployment environment at time t and the image p_i in the scene library, as shown in Equation (6). The closer $d(sf\_v_t, ef\_v_i)$ is to 1, the higher the similarity between sf_v_t and ef_v_i.

$$d(sf\_v_t, ef\_v_i) = \frac{\mathbf{sf\_v_t} \cdot \mathbf{ef\_v_i}}{\|\mathbf{sf\_v_t}\| \|\mathbf{ef\_v_i}\|} \tag{6}$$

Hash organization form: The simplest form of representation vector organization is linear organization. The search process is to compare sf_v_t with ef_v_i in the scene library one by one. As shown in Fig. 8b, we perform linear search experiments on [512, 1024, 2048] dimensional vectors with [1000, 5000, 10000, 20000, 50000] images. The result shows that the time use of linear search depends on the image scale of the scene library. As the size of the scene library increases, the linear search time exceeds the win_{ol}. The increase of search time greatly compresses the time of online training, so the image representation vector cannot be organized by linear.

Inspired by the LSH [26] algorithm, we organize the representation vectors in the scene library in the form of hash tables. Through the design of the hash function to achieve fast mapping key to value, the same key value is mapped to the same slot, and the time complexity of the mapping is usually $O(1)$. We use the hash table to organize the representation vector to complete the rapid mapping of the environment descriptor sf_v_t to the matching representation vector. Let's first explain the data insertion method, and then introduce the fast lookup strategy based on Ebv-List.

Real-time data stream insertion method: We use a hash table to store the representation vectors in groups. The hash table consists of N tuples (named as $Tuple\_HT$), and each tuple comprises two elements, represented as $Tuple\_HT(gk_n, [ef\_v_1, ef\_v_2, ..., ef\_v_i, ...]), n \in N$. Here, $[ef\_v_1, ef\_v_2, ..., ef\_v_i, ...]$ is the set of representation vectors stored in this tuple. gf_v_n represents the "key vector" of this tuple. We consider that the representation vectors within a tuple have relatively high similarity, while the similarity between represent vectors in different tuples is relatively low.

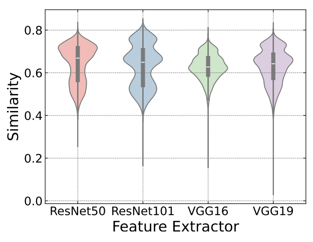
For a newly arrived image p_i, according to Equation (3), the Env-Feature Extractor first obtains its feature vector ef_v_i, and then the Key Generator uses the matrix $\mathbf{W} \in \mathbb{R}^{H \times V}$ ($\mathbf{W}$ is a random Gaussian matrix) to obtain $ef\_v_i{}'$, which is expressed as Equation (7):

$$ef\_v_i{}' = \mathbf{W} \cdot (ef\_v_i)^T \tag{7}$$

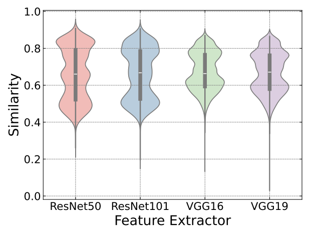
where $ef\_v_i{}' = [e_i^0, e_i^1, ..., e_i^j, ...]$, with e_i^j being a float number. Next, the Key Generator uses Equation (8) to encode $ef\_v_i{}'$ and obtain the key value ek_i corresponding to the image p_i.

$$ek_i = \begin{cases} 1 & \text{if } e_i^j \geq 0 \\ 0 & \text{if } e_i^j < 0 \end{cases} \tag{8}$$

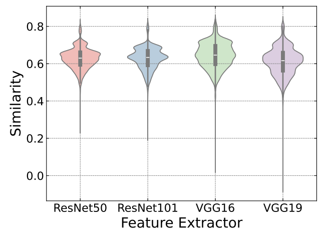
When ek_i is the same as gk_n, we insert ef_v_i into the tuple. The insertion method places relatively similar representation vector into the same set, similar to a grouping strategy. This method supports offline insertion and also supports online insertion. By using the aforementioned real-time data stream insertion method, scene library can update images online, effectively expanding the variety of scenes. Next, we will explain the fast lookup strategy.



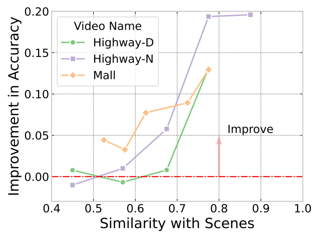
(a) Highway-D



(b) Highway-N



(c) Mall



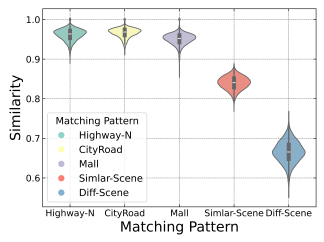
(d) F1-score and similarity

Fig. 9: Relationship between similarity and F1-score. The subfigures (a), (b), and (c) illustrate the impact of different extractors on the distribution range of similarity. The subfigure (d) demonstrates the positive correlation between similarity and the improvement in F1-score of continuous learning.

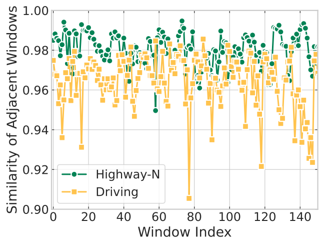
D. Speedup Strategy Based on Ebv-List

The search process needs to be mapped to the $Tuple\_HT$ according to the environment descriptor, and then the set of representation vectors in the hash table is compared one by one. Because the comparison method is linear search, which consumes more time (refer to Fig. 8b for linear search speed). Therefore, we should develop a strategy for accelerating lookups.

Intuitively, the more similar the images used for online training, the more accurate the updated model will be. We conduct experiments on three datasets to test the similarity distribution and the influence of similarity on the F1-score improvement after online training. As shown in Fig. 9a, Fig. 9b, and Fig. 9c, we use ResNet50 [27], ResNet101 [27], VGG16 [28], VGG19 [28] to extract the features of all the images in the scene library. The similarity is calculated by using the data in Highway-D [23], Highway-N [24], Mall [10] scenes and the data in the scene library. Experimental



(a) Distribution of different matching patterns



(b) Adjacent window similarity matching

Fig. 10: Similarity distribution. The Fig. 10a illustrates the similarity of frames within the same video and the dissimilarity of frames from different scenes. Fig. 10b highlights the similarity between adjacent frames. Based on the video temporal correlation, we designed the Ebv-List algorithm.

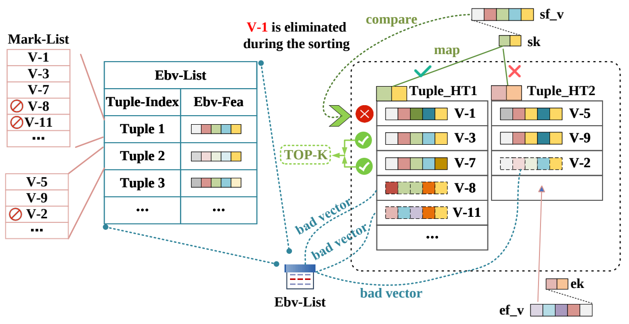


Fig. 11: Accelerated search strategy based on Ebv-List.

results show that the similarity is mainly distributed around 0.4-0.9. The similarity distribution varies depending on the feature extractor, but the range is relatively similar.

Based on the distribution of similarity ranges, we use the feature vectors extracted by the ResNet50 feature extraction part to calculate the relationship between similarity and F1-score after continuous learning. As shown in Fig. 9d, the vertical axis represents the proportion of F1-score improvement after continuous learning with respect to Edge-Only mode. The F1-score of lightweight models will be reduced if they are trained with images with low similarity to the deployment environment, as with scenes like Highway-N and Highway-D, where the similarity is below 0.6. There is a generally positive correlation between similarity and improved F1-score after continuous learning.

Video temporal correlation: Videos have temporal correlation, which is the similarity between frames. As shown in Fig. 10a, we calculate the similarity distribution between frames within the Highway-N, City-Road, and Mall datasets, the similarity distribution between frames of similar scenes, and the similarity distribution between frames of different scenes. Experimental results show that the similarity of frames in the same scene is mostly above 0.9, and the similarity of similar scenes is between 0.8 and 0.9. We measured the similarity of frames between two adjacent windows (window length is 1s), and the similarity between frames of adjacent windows taken by a stationary camera (Highway-D) is higher than that of the autonomous driving scenario (Driving [29]). However, the similarity between adjacent frames is mostly concentrated above 0.94.

Ebv-List: Based on the above two findings, we have designed a data structure called Ebv-List (List of Excluding Bad Vectors) and a fast search algorithm. By using the temporal correlation of videos, we can reduce the number of comparisons in the $Tuple\_HT$ of hash table when searching for similar scenes. As shown in Fig. 11, the Ebv-List (in Fea-Cache) stored the search records of $Tuple\_HT$. The number of $Tuple\_HT$ in the hash table is consistent with the number of records in the Ebv-List. Each record can be represented as a tuple $Tuple\_Ebv$ $(Tuple\text{-}Index, Ebv\text{-}Fea, Mark\text{-}List)$, where $Tuple\text{-}Index$ indicates the $Tuple\_HT$ in the hash table, $Ebv\text{-}Fea$ stores the the latest environment descriptor sf_v mapped to this $Tuple\_HT$, and $Mark\text{-}List$ stores the records of the sequential search for sf_v. If the calculated similarity is less than the $Vector\ Similarity\ Threshold$ (VST), the $Mark\text{-}List$ record is set to False; otherwise, it is set to True. When the similarity is below the VST, we consider the representation feature corresponding to the image having no value for continuous learning, so we refer to such a feature vector as a "bad vector". Due to the temporal correlation of the video, there is a high probability that the next sf_v mapping to this tuple will have very close comparison results. Next, we will explain the specific search process.

The search algorithm can be obtained from the Algorithm 1. At the start of win_{ol}, CL-Shield captures a frame from the buffer, converts it in the Env-Sensor to obtain sf_v_t, and then sends it to the Cloud. In the cloud, the key generator transforms sf_v_t into a key value sk_t, following the same process as the data stream insertion method. Next, sk_t is used to map to the corresponding $Tuple\_HT_k$ in the hash table. Ebv-List is queried to obtain the $Ebv\text{-}Fea_k$ and $Mark\text{-}List_k$ corresponding to $Tuple\_HT_k$. The similarity $sim\_compare$ between sf_v_t and $Ebv\text{-}Fea_k$ is calculated. When $sim\_compare \geq Time\ Similarity\ Threshold$ (TST), CL-Shield decides to use the indication of $Mark\text{-}List_k$ for linear comparison of the representation vectors in the $Tuple\_HT$ and does not calculate the similarity with bad vectors. Using the above steps, we exclude some representation vectors (bad vectors), thereby saving time during comparison.

Ebv-List update: Ebv-List needs to be updated according to the changes in the video captured by the sensor. In order to save time for updates, CL-Shield only updates $Mark\text{-}List_k$ when $Ebv\text{-}Fea_k$ becomes invalid. The criterion for invalidation is $sim\_compare < TST$. When $Ebv\text{-}Fea_k$ becomes invalid, CL-Shield compares sf_v_t with all representation vectors in $Tuple\_HT_k$ and re-record $Mark\text{-}List_k$, and then replaces the original storage of $Ebv\text{-}Fea_k$ with sf_v_t. In this way, Ebv-List can be updated according to changes in the video to ensure that the Ebv-List always indicates the most advantageous representation vectors for accuracy improvement.

IV. Experimental setup

This section describes the deployment of the system, the dataset, and the baseline.

Algorithm 1: Search algorithm based on Ebv-List

```
Data: Frames collected by edge devices
Result: Train set $s_w$ for online learning
1  for $win_{ol}^0, win_{ol}^1, ..., win_{ol}^t, ...$ do
2      Send $sf\_v_t$ to the cloud
3      Convert $sf\_v_t$ to the search key $sk_t$
4      Map from $sk_t$ to the $Tuple\_HT_k$
5      Obtain $(Tuple\text{-}Index_k, Ebv\text{-}Fea_k, Mark\text{-}List_k)$
6      $sim\_compare \leftarrow d(sk_t, Ebv\text{-}Fea_k)$
7      if $sim\_compare \geq TST$ then
8          Find in $Tuple\_HT_k$ based on the $Mark\text{-}List_k$
9      else
10         Find in $Tuple\_HT_k$ based on linear search
11         $Ebv\text{-}Fea_k \leftarrow sf\_v_t$
12         Update $Mark\text{-}List_k$
13     end
14     Sort and get Top-K images as $s_t$
15 end
```

A. System Deployment

Hardware and software: We utilize Python to implement CL-Shield, and deploy the system on an Ubuntu 18.04 machine using both edge and cloud. We conduct DNN model inference and training process on the GeForce RTX 2080 SUPER GPU. The deep learning model is accomplished through PyTorch [30], which incorporates a real-time inference process for the lightweight model, online learning process and inference of "Golden model".

DNN model: The DNN model used in this paper contains an object detection model and a feature extraction model. The object detection model is responsible for inference. We use EfficientDet-D0 [31] as the Lw-Model, and we use EfficientDet-D7 [31] as the "Golden Model" to provide labels for online training. Four classic feature extractors, ResNet50 [27], ResNet101 [27], VGG16 [28] and VGG19 [28], were used as the Env-Sensor, respectively. We keep the final feature output of ResNet, and adjust the VGG feature output from 4096 to 2048 dimensions. Below we list the information of these four feature extractors in Table II.

Metrics: We have established two metrics to evaluate performance: F1-score and latency (in seconds).

TABLE II: Details of Env-Sensor

Model name	Output layer name	Dim	Final dim
ResNet50 [27]	AvgPool	2048	2048
ResNet101 [27]	AvgPool	2048	2048
VGG16 [28]	FC1	4096	2048
VGG19 [28]	FC1	4096	2048

B. Dataset

The data set is divided into the data set for testing and the data set that makes up the scene library. In order to remove the influence of the object detector, the "Ground Truth" that we verify the F1-score on the test set is from EfficientDet-D7 [31]. It is important to emphasize that the scene in the test data set is different from the scene library.

TABLE III: Test dataset

Video name	Source	Total duration
Highway-D	YouTube [23]	600 seconds
Highway-N	YouTube [24]	600 seconds
Mall	Mall dataset [10]	1000 seconds
Driving	YouTube [29]	1200 seconds
City	YouTube [32]	10 hours

TABLE IV: Scene library data source

Scene name	Data source	Label source
Park	Open-Source [20]	Golden Model
City-Road	YouTube [21]	Golden Model
Mot20	Open-Source [22]	Artificial
COCO2017	Open-Source [25]	Artificial
Driving	YouTube [33]	Golden Model
Bit	Open-Source [34]	Artificial

Video set for test: The test scenes we used in this paper could leak user privacy, such as traffic scenes (leaked license plates and driving trajectory), shopping malls (leaked face information, walking trajectory, pedestrian pose), and driving (users do not wish to upload their own videos). Due to ethical considerations, we use the dataset from YouTube and open-source. We did not utilize any home camera footage for testing. Please refer to Table III for dataset specifics.

Video set in scene library: We have used datasets from diverse scenes for CL-Shield's scene library, including video data captured by natural cameras and open-source datasets. The labels are primarily artificial labels from the dataset itself or "Golden Model" (EfficientDet-D7). We ensure a balanced number of images for different scenes in constructing the scene library. Refer to Table IV for specific dataset details.

C. Baseline

We select a number of algorithms that serve as baselines in our experiment. We ensure that the settings are consistent in baselines which has an online learning process, such as the same number of images and learning rate.

1) Edge-Only : A pre-trained lightweight model is run on the Edge device without cloud assistance. In this mode, we will get the original inference F1-score of the lightweight model.

2) AMS [8]: AMS is a classic algorithm commonly used in continuous learning systems, primarily geared towards semantic segmentation tasks. In our case, we have adapted it for object detection tasks. The online learning window of AMS is set at 10 seconds, during which one frame is captured per second from the real-world deployment environment and then uploaded to the cloud server for online training. The training labels are only obtained from the "Golden Model". In this paper, we consider the F1-score achieved by AMS as the upper limit of the continuous learning system, without taking privacy protection into account. We update all parameters of the student model, and the student and teacher models are EfficientDet-D0 [31] and EfficientDet-D7 [31], respectively.

3) EAF [6]: This is a continuous learning system algorithm. The original paper does not provide a name for this system, so we will name it the Edge-Assisted Framework Algorithm

(EAF). EAF selects key frames to send to the cloud server by calculating the proportion of low-confidence objects at the edge and reduces the number of training frames sent by calculating frame redundancy. EAF also decides whether to start the continuous learning process by evaluating fluctuations in accuracy. EAF does not consider privacy protection issues.

4) CL-Blur: We blur images taken from the deployment scene using a Gaussian kernel and then upload them to the cloud server for online training. The labels used for online training are provided by the "Golden Model".

5) Ours (CL-Shield): Follow Fig. 7, we deploy a distributed continuous learning system. win_{ol} is set to 10s and we set the system super-parameters $VST = 0.65$ and $TST = 0.925$. The values of the super-parameters in the CL-Shield are discussed in the experiment results. The only difference between AMS and CL-Shield is that AMS uses fully real-world data from the deployment environment, while CL-Shield trains using the scene library.

6) Ours-Random: This mode randomly selects K frames from the scene library as training sets in a win_{ol} instead of the most similar frames, and the rest of the operations are consistent with CL-Shield.

7) FG-Removal: This pattern identifies and removes common foreground objects. For the static camera scenes (Mall, Highway-D, Highway-N), we deploy the KNN-based background modeling algorithm [19] on the edge, which only sends the established background frames (as training sets) to the cloud. For the dynamic scene camera scene (Driving), the background modeling algorithm is not effective, we deploy the model segmentation algorithm (Deeplabv3-ResNet101 [18]) on the edge to detect and mask foreground objects. Then the processed images are sent to the cloud to participate in the online training process. While FG-Removal can identify and hide most foreground objects, it is ineffective at removing background privacy.

V. Experiment Results

We conduct relevant experiments to verify the performance of the system. The system will be tested to determine whether it can meet the two goals of privacy protection and F1-score improvement. Next, we discuss the parameter values of the system. Finally, we test the forgetting of the system.

A. Overall Performance

The F1-score comparison results between CL-Shield and the corresponding baselines are shown in Fig. 12, and the performance comparison is shown in Table V.

Edge-Only: Deploying a lightweight detection model only on edge can avoid user data transmission and privacy leakage. The disadvantage is that data drift issues may occur, which can decrease the F1-score of video analytics.

AMS: Its F1-score can be improved by 18.22%-65.63% compared to Edge-Only mode, achieving the highest F1-score among all baselines. A disadvantage is that privacy is not protected.

EAF: EAF intermittently performs online training on the cloud, resulting in accuracy that is higher than the Edge-Only

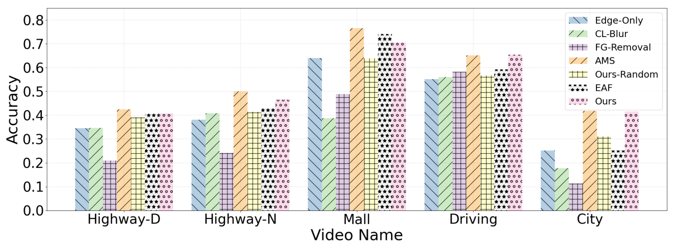


Fig. 12: Overall performance.

mode but lower than the AMS algorithm (the upper limit of continuous learning accuracy). The accuracy of the EAF mode is not necessarily superior to that of CL-Shield. Additionally, EAF directly uploads images collected from edge devices to the cloud server, posing a higher risk of user privacy leakage.

CL-Blur: The F1-score of CL-Blur has achieved a slight improvement on the Highway-N and Driving datasets, but its F1-score on the Mall dataset is significantly lower than Edge-Only mode. The reason is that the objects in the former dataset are larger, and the degree of blurring is weaker, so the "Golden Model" can still identify some objects and give labels. With the same degree of blurring, the objects in the mall are smaller than Highway-D, so all target features in the mall dataset will be lost. Therefore, this mode is limited, and it cannot accurately estimate the relationship between the degree of blurring and the effectiveness of privacy removal.

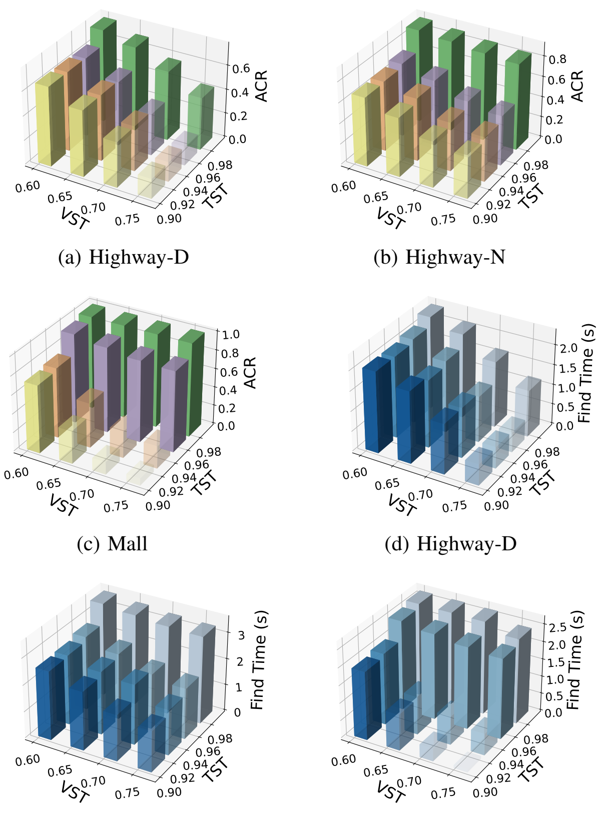
FG-Removal: The background modeling algorithm is applied in scenes with static backgrounds, and can effectively remove all moving objects (most foreground targets). The F1-score after online learning is significantly lower than that of Edge-Only mode. The segmentation-based algorithm effectively eliminates the majority of foreground targets. However, the small number of remaining targets has a limited impact on improving the F1-score of the continuous learning system, resulting in an increase of only 5.56% on the Driving dataset. Therefore, this removal of foreground mode cannot be directly used in continuous learning systems. System F1-score is negatively affected by privacy leaks involving foreground targets and backgrounds.

Ours: The algorithm proposed in this paper demonstrates significant improvements in F1-score, achieving increases of 18.01%, 22.19%, 10.25%, 17.94% and 64.74% on the Edge-Only dataset. Furthermore, it attains F1-score rates of 96.07%, 93.05%, 92.1%, 99.77% and 99.45% on five respective datasets, in comparison to the AMS F1-score. CL-Shield effectively prevents foreground and background information leakage by simply adding a lightweight model on the edge device, thus relieving the computational burden. Consequently, CL-Shield successfully accomplishes the objectives of enhancing F1-score and providing comprehensive privacy protection, as outlined in this paper.

Ours-Random: Randomly selecting images from the scene library for training introduces a high degree of chance, leading to a slight decrease in F1-score on the Mall dataset. While this approach may prevent user information leakage, it also introduces instability to the system's F1-score, making it unsuitable for practical use.

TABLE V: Baseline performance comparison

Method	F1-score improvement	Protect foreground	Protect background	Additional burden on the edge
Edge-Only	✖	✓	✓	None
AMS	✓	✖	✖	None
CL-Blur	maybe	✓	✓	Gaussian Kernel
Ours-Random	maybe	✓	✓	None
FG-Removal	maybe	maybe	✖	Background extraction model Semantic segmentation model
EAF	✓	✖	✖	Keyframe detection algorithm
Ours	✓	✓	✓	Lightweight model



(a) Highway-D (b) Highway-N (c) Mall (d) Highway-D (e) Highway-N (f) Mall

Fig. 13: The effect of the parameters of Ebv-List on the contrast ratio and find time.

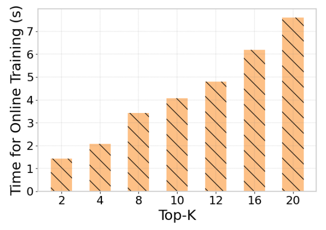
B. Sensitivity Testing of Online Module

In Section III, we set several hyperparameters for the system, including the Top-K number of images involved in one-time online training, the VST and the TST for accelerating the search. Generally, these parameters determine how images are selected and how long training takes online. We analyze their specific impacts as follows.

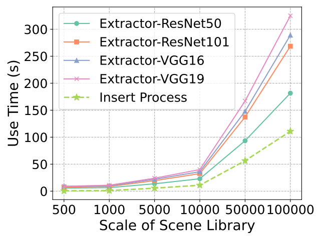
1) Top-K: Top-K is used to select the most similar K images. The larger the value of K, the longer time used for online training. We conducted tests on two GeForce RTX 2080 SUPER GPUs, and the results are shown in Fig. 14a. In order to execute processes such as finding similar images and online training within the continuous learning window win_{ol}, we set Top-K = 12.

2) Parameters in Ebv-List: In the design of Ebv-List, the $Vector\ Similarity\ Threshold$ (VST) controls the number of representation vectors involved in the comparison, and the $Time\ Similarity\ Threshold$ (TST) controls the frequency of updating Ebv-List. These two parameters play an important role in the matching of representation vectors. Both parameters are intuitively related to video features and to scene library quality. Keeping the scene library consistent, we conduct experiments on the Highway-D, Highway-N, and Mall datasets, jointly analyzing the impact of these two parameters on the matching of similar images within a continuous learning window. Here, we provide a method to measure the performance of Ebv-List - Average Contrast Ratio (ACR), calculated as the ratio of the number of comparisons using Ebv-List to the number of comparisons without Ebv-List when searching for the Top-K representation vectors. A smaller ACR value indicates fewer comparison times and a shorter average search time (denoted as Find Time in the figure) when searching for Top-K representation vectors. Fig. 13 illustrates the average contrast ratio and the average find time. Experimental results demonstrate that Ebv-List saves time in finding similar environments, and as TST increases and VST decreases, the number of comparisons decreases, leading to less time spent, which is consistent with the design logic.

Offline module sensitivity test: Two processes are involved in establishing the scene library: (1) generating representation vectors for new images (extracting process) and (2) inserting the vector into the hash table (insert process). Both processes are carried out offline while supporting online operations. In this section, we discuss the performance of the extracting and inserting process. We set the scale of the scene library to [500, 1000, 5000, 10000, 50000, 100000] images, and use ResNet50, ResNet101, VGG16, and VGG19 as the feature extractors in Env-Feature Extractor. The information on these feature extraction networks is shown in Table II. Experimental results are shown in Fig. 14b. As the scale of the scene library increases, the time required for generating vectors increases, ranging from a few seconds to several hundred seconds. Meanwhile, the time required to insert data streams into the Hash Table also increases. The establishment time of a scene library at a level of 100,000 images (including the process of establishing representation vectors and organizing using the Hash Table) can also be completed within 10 minutes



(a) Impact of Top-K on online training time



(b) Build speed

Fig. 14: System module running time sensitivity experiments. Fig. 14a shows the impact of Top-K on time for online training, and Fig. 14b displays the impact of scene library scale on use time under various feature extractors.

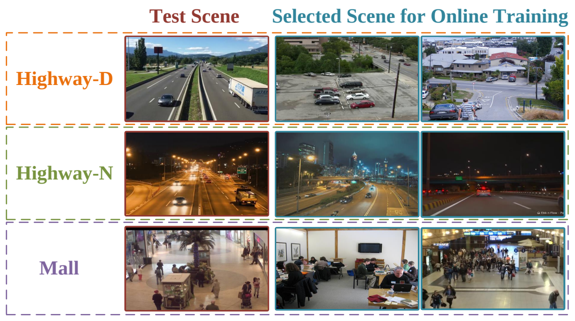


Fig. 15: Find similar scenes.

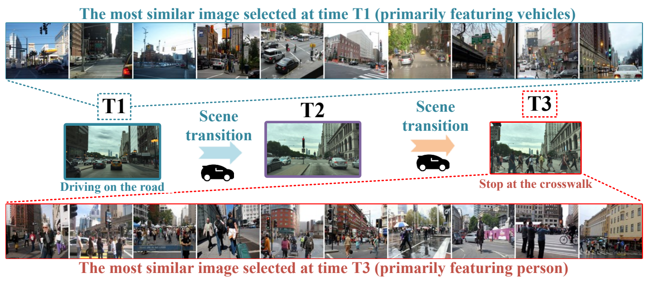

The most similar image selected at time T1 (primarily featuring vehicles)
T1
Scene transition
T2
Scene transition
T3
Driving on the road
Stop at the crosswalk
The most similar image selected at time T3 (primarily featuring person)

Fig. 16: CL-shield selects appropriate images for training when deploying scene transitions.

in these offline scenarios.

C. Visualization of Operation

1) The most similar images selected: CL-Shield selects similar scenes from the scene library for online training, and whether to select similar images has a great impact on the final F1-score of continuous learning. As shown in Fig. 15, we show the scenes that are selected more frequently in continuous learning. We can see that for Highway-D, CL-Shield often chooses scenarios with more vehicles. For the Highway-N scene, CL-Shield often selects those driving scenes at night, because light intensity has a significant impact on scene similarity. For the data set Mall, there are usually a lot of people in the scenes selected by CL-Shield, because Mall is an indoor crowded scene. The experimental results show that CL-Shield can select some scenes that are beneficial to its F1-score improvement, which proves that our system is effective.

2) Scene transition: To test whether CL-Shield can select appropriate images for training during scene transitions, we conducted an experiment on scene changes. As shown in Fig. 16, we used a continuous autonomous driving video for testing and filled the scene library with the COCO 2017 train dataset [25]. At time T1, the vehicle is driving on the road, and the objects captured by the camera are primarily vehicles. At time T2, a crosswalk appears ahead of the vehicle. At time T3, the vehicle is stopped in front of the crosswalk, and the objects captured by the camera are primarily pedestrians. As needed, the system selected more vehicle-centered images at T1 (the top row of images). As shown in the bottom row of images, at T3, the system selected pedestrian-centered images (some of which also include the crosswalk). This experiment validates that our system can select images most similar to the deployment environment during scene transitions for training.

D. The Influence of Images of Same Class

In this section, we aim to test the reasons for the improvement in continuous learning accuracy. Is it due to: (1) training with objects of the same class as those in the deployment environment, or (2) training with images similar to the deployment environment? We filtered images containing “vehicles” and images containing “person” from the COCO2017 [25] training set, naming them Class-Based in Fig. 17. The filtering method primarily relied on the labels provided by the COCO2017. Then, we trained the model using images containing ”vehicles” in each online training window and tested it on the “Highway-D” and “Highway-N”, as the main objects captured in the “Highway-D” and “Highway-N” datasets are vehicles. Similarly, we used images containing ”person” to train the model and tested it on the “Mall” dataset. It is important to emphasize that the number of training images in each training window is the same for all experimental groups.

As shown in Fig. 17, selecting only images of the same class (but not similar) for training did not achieve better performance than the Edge-Only mode on the Highway-D and Highway-N datasets, indicating that it does not have a

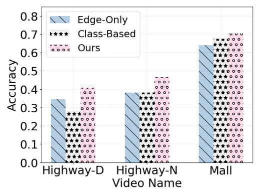


Fig. 17: The influence of images of same class. The left subfigure shows that using images of the same class as the deployment environment (where the same category refers to the foreground objects) for continuous learning may result in lower accuracy than the Edge-Only mode. The right subfigure displays images of the same class selected from the COCO2017 dataset for continuous learning.

positive effect on accuracy. By analyzing the images used in training, we identify two reasons: (1) both foreground objects and backgrounds can provide the necessary information for training, and (2) the training set of objects from the same class also includes a variety of complex features, making it challenging for the object detection network to train effectively on these complex features. This experiment demonstrates the rationale behind choosing similar images for training rather than only selecting images of the same class. It also indicates that the improvement in accuracy of CL-shield does not come from simply increasing the number of training epochs.

E. System Time-Based Performance

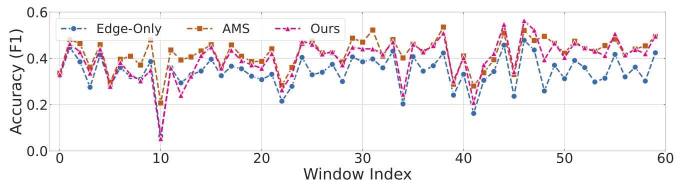
In this section, we present the F1-score variation over different continuous learning windows. Fig. 18 presents the F1-score changes of Edge-Only, AMS, and our models on the four datasets with varying time windows. Results reveal that CL-Shield achieves a window F1-score above Edge-Only in 83.33%, 100%, 85%, and 87.5% of the windows across the four datasets, and it can rapidly adapt to scene changes. CL-Shield achieves a window F1-score greater than 90% of AMS F1-score in 81.67%, 61.67%, 67%, and 94.17% of the windows across the four datasets. Combined with Section V-A, it can be observed that CL-Shield not only exhibits a significant improvement in overall F1-score but also demonstrates high F1-score stability over the time axis.

F. System Forgetting Test

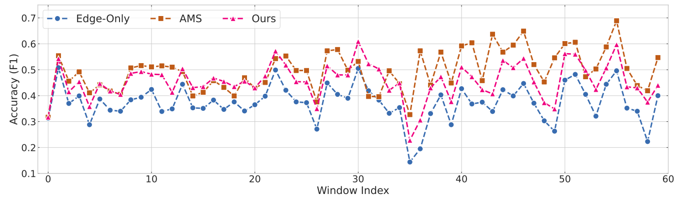
We collected a video dataset called city [32] to test the possibility of a decrease in F1-score during the runtime of the system. The city video data is a continuous 10-hour video. We demonstrated the performance of three modes, Edge-Only, AMS, and ours, in this video in the time windows [660, 720], [1380, 1440], [2100, 2160], [2820, 2880], and [3480, 3600]. The experiments in Fig. 19 showed that even in a sequential 10-hour video, the F1-score of our designed system remained stable around the AMS mode, and there was no decrease in F1-score due to forgetting or overfitting in similar scenes. Our system exhibits operational robustness.

VI. Related Work

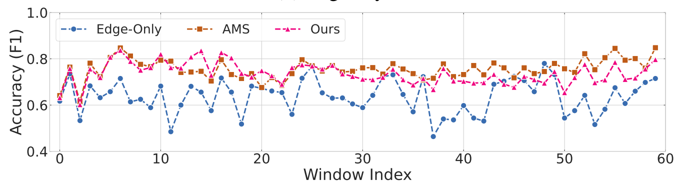
Video analytics system: The video analytics system [1], [35], [36] utilizes DNN to analyze videos captured at the edge. Among the most common methods used by the video analytics system are GPU resource allocation, transmission configuration optimization, and model splitting. While AWStream [2] and DDS [3] adjust the video configuration sent to cloud servers to adapt to varying bandwidths, the current bandwidth in the deployment environment is no longer a bottleneck for video analytics system development. Neurosurgeon [37] unloads most of the computation to cloud servers through model splitting; however, model splitting is only suitable for image classification models with relatively simple topological structures. The current challenges in deploying the video analytics system revolve around the generalization of models and privacy leaks.



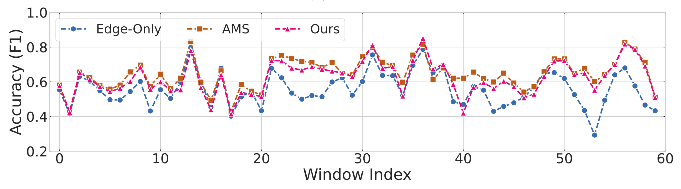
(a) Highway-D



(b) Highway-N

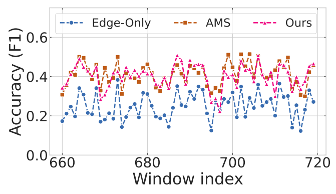


(c) Mall

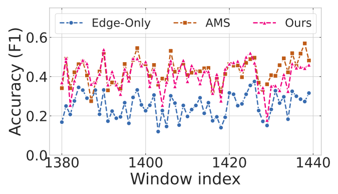


(d) Driving

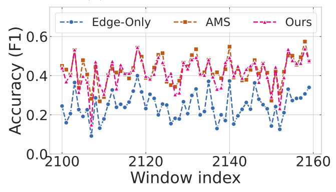
Fig. 18: Sub-window F1-score performance.



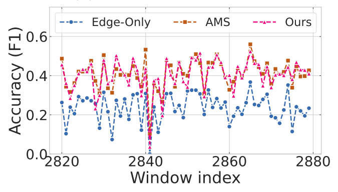
(a) Win660-Win720



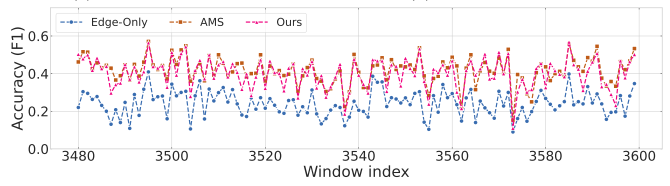
(b) Win1380-Win1440



(c) Win2100-Win2160



(d) Win2820-Win2880



(e) Win3480-Win3600

Fig. 19: Forgetting test.

Continuous learning system: The actual deployment scenarios of cameras are often complex and variable, differing from the distribution of the model's offline training data. This leads to lower accuracy of lightweight models during real-time inference (data drift). A continuous learning system can enhance inference accuracy during deployment by collecting images from the deployment environment for online model training. Mullapudi R T et al. [38] design lightweight student models suitable for training on edge devices, leading to significant advancements in semantic segmentation tasks. However, their work does not account for the constraints encountered in practical deployment scenarios. In response to this, Mehrdad Khani et al. [8] introduce the AMS algorithm, a continuous learning system simulation based on edge-cloud architecture for semantic segmentation tasks. AMS deploys the student model on edge devices for real-time video inference tasks, while a teacher model is hosted on a cloud server to provide labels and conduct online training of the student model. Furthermore, Bhardwaj R et al. [7] address the resource competition scenario in continuous learning systems on edge servers and propose the Ekya system for GPU space resource allocation. Additionally, RECL [9] establishes a model zoo on edge cloud to facilitate rapid online training and updates of models, storing models trained in various environments to achieve the goal of rapid model training and updates online.

These continuous learning systems [7]–[9] rely on the knowledge distillation process on cloud servers for online training of lightweight models. In knowledge distillation, the teacher model provides labels for training the student model. We have adopted the concept of knowledge distillation, but with some differences from these works [7]–[9]. For CL-Shield, the labels used for online training come from both the teacher model and open-source datasets. The label acquisition process and the online training process execute without interfering with each other in CL-Shield.

In addition, we have noted another subfield in continuous learning: class-incremental learning. Class-incremental learning increases the number of output classes of the model using training data with new classes in the data stream. EWC [15] addresses the issue of model forgetting by protecting important parameters. iCaRL [14] employs a distillation learning mechanism and designs exemplar sets. WA [13] corrects the biased weights in the FC layer during training. Other algorithms [16], [39] expand the model structure, achieving a better balance between model stability and plasticity. However, we do not consider solving the class-incremental problem in this paper.

Privacy preserving algorithm: In the field of privacy protection, the simplest method [17], [40] is to directly apply blur or masking to the privacy-sensitive areas (e.g., pedestrians) in images. Zheng et al. [41] propose a method to hide visual privacy information by shuffling the order of blocks in images. However, this method is only designed for VIT (Vision Transformer) classifiers and is not suitable for complex DNN models such as object detectors. Sun et al. [42] propose a two-stage structure to hide facial privacy information, which can modify faces in natural images and add facial information to blacked-out areas. Tian et al. [12] extract motion information from encrypted video bit streams for object detection and tracking tasks. This algorithm extracts designed features from video bit streams based on the H.265 encoding algorithm, avoiding the decoding of original images on cloud servers. Privid [11] transforms object location into continuous appearance time length information and implements the system with a specific database handling method. However, Privid does not support more detailed queries, such as object location information, which significantly affects the application of the video analytics system. SPNN [43] designs a vertical federated learning algorithm to ensure that two users use the same labels when training a model. However, federated learning algorithms generally consider only joint training of models by multiple users, which is significantly different from continuous learning systems (joint inference and training processes). In conclusion, there is no algorithm specifically designed to address privacy leakage in continuous learning systems.

VII. DISCUSSION

A. Scene library expansion

A core component of CL-Shield is the scene library. The ability of the continuous learning system to match appropriate scene images for online training is contingent upon the size of this library. An increased library size enhances the probability of matching scenes similar to the deployment environment. We incorporated real-time data stream insertion method and two data sources (open-source data and unlabeled data) into the system design, providing a technical foundation for online database expansion. To minimize the negative impact of library size, continuous learning systems in similar deployment contexts can share the huge scene library. The discussion of sharing methods is left for the future.

B. Similar image selection method

We use similar vectors to select images that are comparable to the deployment environment for training, which is a relatively straightforward approach. However, to protect user privacy, based on our current knowledge, we can only compress image information into vector information, which may result in some loss of information. In the future, we can enhance feature matching by gathering additional environmental information from edge devices such as the time of capture, weather conditions, and so on.

VIII. CONCLUSION

This paper presents CL-Shield, a privacy-protected continuous learning system. We introduce an integration of a dynamically updated scene library in the continuous learning system, housing images depicting diverse scenarios. The proposed approach eliminates the need for direct image acquisition from the operational environment during online training, thereby ensuring robust user privacy protection. Furthermore, we have devised a swift retrieval mechanism utilizing Ebv-List, facilitating prompt identification of the optimal training images within a mere 2 seconds. Our experiments have demonstrated that CL-Shield can achieve an F1-score of over 92% in continuous learning systems without privacy protection, while also ensuring no instances of forgetting.

Acknowledgments

This work is supported by the Major Key Project of PCL under grant No. PCL2023A06, the National Key Research and Development Program of China under grant No. 2022YFB3105000, and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989.

References

[1] K. Du, Q. Zhang, A. Arapin, H. Wang, Z. Xia, and J. Jiang, "Accmpeg: Optimizing video encoding for video analytics," *arXiv preprint arXiv:2204.12534*, 2022.

[2] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 236–252.

[3] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 557–570.

[4] M. Zhang, F. Wang, and J. Liu, "Casva: Configuration-adaptive streaming for live video analytics," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 2168–2177.

[5] J. Yi, S. Choi, and Y. Lee, "Eagleeye: Wearable camera-based person identification in crowded urban spaces," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[6] Y. Kong, P. Yang, and Y. Cheng, "Edge-assisted on-device model update for video analytics in adverse environments," in *Proceedings of the 31st ACM International Conference on Multimedia*, ser. MM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 9051–9060. [Online]. Available: https://doi.org/10.1145/3581783.3612585

[7] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, N. Karianakis, Y. Shu, K. Hsieh, V. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," *arXiv preprint arXiv:2012.10557*, 2020.

[8] M. Khani, P. Hamadanian, A. Nasr-Esfahany, and M. Alizadeh, "Real-time video inference on edge devices via adaptive model streaming," *arXiv preprint arXiv:2006.06628*, 2020.

[9] M. Khani, G. Ananthanarayanan, K. Hsieh, J. Jiang, R. Netravali, Y. Shu, M. Alizadeh, and V. Bahl, "RECL: Responsive Resource-Efficient continuous learning for video analytics," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 917–932. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/khani

[10] C. C. Loy, S. Gong, and T. Xiang, "From semi-supervised to transfer counting of crowds," in *IEEE International Conference on Computer Vision*, 2014.

[11] F. Cangialosi, N. Agarwal, V. Arun, S. Narayana, A. Sarwate, and R. Netravali, "Privid: practical,{Privacy-Preserving} video analytics queries," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 209–228.

[12] X. Tian, P. Zheng, and J. Huang, "Robust privacy-preserving motion detection and object tracking in encrypted streaming video," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5381–5396, 2021.

[13] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia, "Maintaining discrimination and fairness in class incremental learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 13 208–13 217.

[14] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[15] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, pp. 3521 – 3526, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:4704285

[16] S. Yan, J. Xie, and X. He, "Der: Dynamically expandable representation for class incremental learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 3014–3023.

[17] C. Liu, T. Zhu, J. Zhang, and W. Zhou, "Privacy intelligence: A survey on image privacy in online social networks," *ACM Comput. Surv.*, vol. 55, no. 8, dec 2022. [Online]. Available: https://doi.org/10.1145/3547299

[18] L. C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017.

[19] Z. Zivkovic and F. V. D. Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.

[20] X. Wang, E. Swears, A. Hoogs, S. Oh, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, and D. Ramanan, "A large-scale benchmark dataset for event recognition in surveillance video," in *IEEE International Conference on Advanced Video Signal Based Surveillance*, 2011.

[21] Wanderer Ambience, "City traffic heavy rain sounds for sleep," 2020, https://www.youtube.com/watch?v=dVB0dN6Zmfo&list=PLV6xYW_d-fzffhX_h9tp9slX_nWRTmL9k, Last accessed on 2020-10-13.

[22] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv*, 2020.

[23] YouTube, "Relaxing highway traffic," https://www.youtube.com/watch?v=nt3D26lrkho, 2017, accessed on 2017-7-4.

[24] YouTUbe, "Asmr/highway," https://www.youtube.com/watch?v=xEtM1I1Afhc, 2019, accessed: 2019-1-30.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[26] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, ser. VLDB '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, p. 518–529.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[29] Seoul Walker, "Night driving seoul city — gangnam and expressway with chill lofi hiphop beats pov 4k hdr," 2022, https://www.youtube.com/watch?v=40xZVEFVBuE, Last accessed on 2022-05-28.

[30] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[31] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[32] CITY TRAFFIC, "Road atmosphere at night," 2021, https://www.youtube.com/watch?v=KRYTS2UW9S4, Last accessed on 2021-06-10.

[33] Seoul Walker, "Seoul night drive," 2022, https://www.youtube.com/watch?v=SRpMapyw6Aw, Last accessed on 2022-08-18.

[34] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle type classification using a semisupervised convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2247–2256, 2015.

[35] S. Liu, T. Wang, J. Li, D. Sun, M. Srivastava, and T. Abdelzaher, "Adamask: Enabling machine-centric video streaming with adaptive frame masking for dnn inference offloading," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 3035–3044.

[36] L. Zhang, Y. Zhang, X. Wu, F. Wang, L. Cui, Z. Wang, and J. Liu, "Batch adaptative streaming for video analytics," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 2158–2167.

[37] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, L. Tang, and C. Lab, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *Acm Sigplan Notices*, vol. 52, no. 1, pp. 615–629, 2017.

[38] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *computer*, vol. 50, no. 10, pp. 58–67, 2017.

[39] D. Kim and B. Han, "On the stability-plasticity dilemma of class-incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20 196–20 204.

[40] J. R. Padilla-López, A. A. Chaaraoui, and F. Flórez-Revuelta, "Visual privacy protection methods: A survey," *Expert Systems with*

Applications, vol. 42, no. 9, pp. 4177–4195, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417415000561

[41] Z. Qi, A. MaungMaung, Y. Kinoshita, and H. Kiya, "Privacy-preserving image classification using vision transformer," 2022.

[42] Q. Sun, L. Ma, S. Joon Oh, L. V. Gool, B. Schiele, and M. Fritz, "Natural and effective obfuscation by head inpainting," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5050–5059.

[43] J. Zhou, L. Zheng, C. Chen, Y. Wang, X. Zheng, B. Wu, C. Chen, L. Wang, and J. Yin, "Towards scalable and privacy-preserving deep neural network via algorithmic-cryptographic co-design," 2022.

Tianyu Li received the B.S. degree in computer science and technology from Anhui University, China, in 2021, and the M.S. degree in computer technology from Tsinghua University, China, in 2024. His main research interests include video analytics, deep learning, edge computing and cloud computing.

Hanling Wang received the B.S. degree in electronic information science and technology from Central South University, China, in 2017, and the M.S. degree in data science and information technology from Tsinghua University, China in 2020. He is currently pursuing the Ph.D. degree in computer science and technology with Tsinghua University (joint program with Pengcheng Laboratory), China. His main research interests include video analytics, edge computing, semantic communication, and deep learning.

Qing Li (Senior Member, IEEE) received the B.S. degree in computer science and technology from Dalian University of Technology, Dalian, China, in 2008, the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2013. He is currently an Associate Researcher with Pengcheng Laboratory, Shenzhen, China. His research interests include network function virtualization, in-network caching/computing, intelligent self-running network, edge computing, etc.

Yong Jiang (Member, IEEE) received the B.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 1998 and 2002, respectively. He is currently a full professor with Division of Information Science and Technology, Tsinghua Shenzhen International Graduate School, Shenzhen, China and Department of Mathematics and Theories, Pengcheng Laboratory, Shenzhen, China. He mainly focuses on the future Internet architecture, Internet of Things, edge computing, AI for networks, etc.

Zhenhui Yuan is an Assistant Professor at the University of Warwick, UK. He received his B.degree (Software Engineering) at Wuhan University, China, in 2008 and PhD (Electronic Engineering) at Dublin City University, Ireland, in 2012. His research interests lie in communications and networking of sensors, robots and vehicles. He was the postdoc researcher (2012-2014) at Ericsson and Enterprise Ireland and the visiting scholar (2014) at Network Research Lab (led by Prof. Mario Gerla), UCLA, U.S. He was 3GPP delegate (2014-2015) on 5G at Huawei (Shanghai). During 2015-2019, He was the Associate Professor in the Key Lab of RF Circuits and Systems (Ministry of Education) in Hangzhou Dianzi Univeristy, China. He was the co-founder and CTO (2015-2019) at RobSense (Hangzhou) Technology Co.Ltd , a startup that designs and produces UAV flight controller and IoT gateway. RobSense has received over £1m venture investment since 2015. He joined Northumbria University (UK) as the Senior Lecturer in 2020 and University of Warwick (UK) as an Assistant Professor in 2023. He won the best paper awards at IEEE ICCRE 2016 and IEEE BMSB 2014. He served as the Guest Editor in IEEE Network and IEEE IoT-Journal, and the Associate Editor at IEEE ACCESS, IEEE Transactions on Consumer Electronics and Journal. Pervasive Computing & Communications. He is a member of IEEE P1954 on UAV communications.