X-Stream: A Flexible, Adaptive Video Transformer for Privacy-Preserving Video Stream Analytics

Dou Feng[1] Lin Wang[2] Shutong Chen[3] Lingching Tung[1] Fangming Liu*[4,1]
[1]National Engineering Research Center for Big Data Technology and System,
Services Computing Technology and System Lab, Cluster and Grid Computing Lab
School of Computer Science and Technology, Huazhong University of Science and Technology, China
[2]Paderborn University, Germany [3]Guangxi University, China [4]Peng Cheng Laboratory, China

***Abstract*—Video stream analytics (VSA) systems fuel many exciting applications that facilitate people's lives, but also raise critical concerns about exposing too much individuals' privacy. To alleviate these concerns, various frameworks have been presented to enhance the privacy of VSA systems. Yet, existing solutions suffer two limitations: (1) being scenario-customized, thus limiting the generality of adapting to multifarious scenarios, (2) requiring complex, imperative programming, and tedious process, thus largely reducing the usability of such systems. In this paper, we present X-Stream, a privacy-preserving video transformer that achieves flexibility and efficiency for a large variety of VSA tasks. X-Stream features three major novel designs: (1) a declarative query interface that provides a simple yet expressive interface for users to describe both their privacy protection and content exposure requirements, (2) an adaptation mechanism that dynamically selects the most suitable privacy-preserving techniques and their parameters based on the current video context, and (3) an efficient execution engine that incorporates optimizations for multi-task deduplication and inter-frame inference. We implement X-Stream and evaluate it with representative VSA tasks and public video datasets. The results show that X-Stream achieves significantly improved privacy protection quality and performance over the state-of-the-art, while being simple to use.**

***Index Terms*—video privacy, privacy preservation, video stream analytics, declarative query language**

I. Introduction

High-resolution surveillance cameras are now pervasive [1], deployed throughout city streets, schools, and private places for safeguarding. To take advantage of these cameras, several video analytics frameworks have been proposed [2]–[5], ranging from object detection to tracking, e.g., traffic monitoring and abnormal events detection. Typically, a video stream analytics (VSA) system deploys cameras and analyzes the video streams produced by these cameras for different purposes. The video streams for analytics may need to be exposed to entities without permission to access the sensitive information contained in the video streams. For example, at an airport, the infrastructure team may want to leverage VSA to detect crowds for better planning, while sensitive information (e.g., human face) is only accessible by the security team for safety-related analytics tasks. Without credible privacy protection, VSA systems are hard to be widely useful due to continuous criticism on their privacy implications. The EU and some cities in the US have even banned facial recognition systems in public areas while waiting for proper solutions [6], [7].

To address the privacy concern, researchers have proposed several mechanisms for transforming raw videos from cameras into privacy-protected versions, maintaining usability for basic analytics tasks [8]–[12]. These mechanisms use various privacy-preserving techniques, including encryption [12], detection-RoI-based inpainting [13], [14], image-processing-based filtering (e.g., blurring and pixelating), Generative Adversarial Network (GAN)-based cartooning [8], and face de-identification [10], [15], [16]. However, these mechanisms fall short for at least the following three reasons in general: (1) Developers need *full knowledge* of these privacy-preserving techniques to choose the right one for a specific use case. (2) They rely on one static privacy-preserving technique, *lacking generality* to adapt to dynamic scenarios. (3) All these mechanisms require complex, imperative programming across low-level libraries such as OpenCV, Pytorch, and dlib [17], which leads to a *tedious* and *error-prone* development process. These factors largely limit the usability of these privacy protection mechanisms for VSA.

In this paper, we propose X-Stream—a flexible, adaptive video transformer for privacy-preserving VSA, requiring minimal developer efforts. X-Stream introduces PriQL, a declarative query language extending SQL with `PROTECT` and `EXPOSE` keywords. Developers can specify content protection or exposure requirements for analytics in familiar SQL terms, allowing them to focus on video transformation logic rather than the detailed properties of privacy-preserving techniques. X-Stream features a runtime system translating PriQL queries into dataflow graphs with various video operators to meet the protection and exposure requirements and system performance through spatial and temporal redundancy. X-Stream introduces new designs to address the following three main challenges in the runtime system.

How to construct the dataflow graph? PriQL queries only specify the privacy protection goals at a high level, without

*The corresponding author is Fangming Liu (fangminghk@gmail.com).

This work was supported in part by National Key Research & Development (R&D) Plan under grant 2022YFB4501703, the Major Key Project of PCL under Grant PCL2022A05, and the High-level Talent Program of Guangxi University under Grant A3070051017. Lin Wang was supported in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 210487104 - SFB 1053.

details regarding what privacy-preserving techniques to use and how to combine them if multiple techniques are needed. To close the gap, X-Stream classifies the target video content (for protection or exposure) into different types. X-Stream keeps a repository of video operators tailored for protecting the content of each type. So far, X-Stream supports three types namely object, background, and motion; more can be added in the future. By parsing PriQL queries, X-Stream identifies the target content types and selects the appropriate video operators from the corresponding repository. X-Stream then constructs a dataflow graph composing these operators, according to the requirements specified in the queries.

How to enable context-aware adaptation? Unlike current methods employing static privacy-preserving techniques for specific VSA tasks, X-Stream adopts an adaptive approach where the privacy-preserving technique together with its parameters are continuously adapted according to the video context. This is inspired by the observation that different privacy-preserving techniques perform differently in terms of protection quality and computational complexity under varying contexts and there is no one-size-fits-all technique that performs well under all possible contexts. To this end, X-Stream introduces a context analyzer generating protection quality and performance information for video operators under the current context. The content analyzer also provides frame information to determine the operators' parameters for video processing. The runtime system uses this information to choose the best video operators when constructing the dataflow graphs.

How to achieve efficient execution? To achieve efficient execution of the video operators for multiple PriQL queries, X-Stream leverages both spatial and temporal redundancy to reduce the computation needed for video transformation. More specifically, X-Stream merges the same video operators from different dataflow graphs generated for different PriQL queries if these queries also share the same input video stream. This can significantly reduce the number of video operators in scenarios where multiple VSA tasks with different privacy-protection requirements need to be performed with the same input video. Additionally, exploiting temporal correlations among video frames, X-Stream employs a random sampling technique to process only those frames with differences exceeding a predefined threshold to reduce the number of processing frames. These optimizations can largely improve the resource efficiency of the X-Stream runtime system.

We have implemented X-Stream in Python, and evaluated its performance via privacy analysis and comprehensive experiments. X-Stream can be used for a variety of scenarios, e.g., traffic surveillance and indoor monitoring. The privacy-enhanced videos generated by X-Stream can outperform those generated by all the baseline systems in our thorough experiments. Compared with the unmodified raw videos, the protected videos also achieve a comparable VSA performance (w.r.t. analytics quality) in two typical VSA tasks.

In summary, we make the following contributions: After introducing the background on privacy-preserving VSA and motivations for our work in Section II, we

- present X-Stream, a novel video transformer for privacy-preserving VSA with built-in usability, flexibility, and efficiency (Section III and Section IV).
- design a declarative query language, PirQL, which extends SQL with two keywords to allow expressing privacy protection requirements for video transformation with familiar SQL syntax (Section IV-A).
- present a method to map the PriQL parsing results to the most suitable privacy-preserving techniques for each identified video content type adaptively, based on the video context (Section IV-C).
- propose performance optimizations that leverage spatial and temporal correlations to avoid redundant computations (Section V).
- implement X-Stream and evaluate it on two commonly-seen VSA tasks (Section VII). The results show that X-Stream outperforms the state-of-the-art systems.

II. Background and Motivation

A. Privacy-Preserving Video Analytics

In Section I, we highlighted the trade-off between privacy protection and intelligibility maximization in Video Source Analytics (VSA). Video owners share sources with subscribers for analytics tasks like pedestrian counting, creating a dilemma. Privacy protection requires concealing sensitive information, while analytics demands maximum exposure of scenes and objects for enhanced utility. For instance, protecting personal identity (e.g., face and dressing) while fully exposing the body outline ensures accurate pedestrian counting. Traditional methods like encryption may compromise intelligibility, proving inadequate for VSA.

The intelligibility of the video frame indicates how much the visual information is retained, i.e., it suggests precision loss when we do the video analytics tasks. The higher the intelligibility of the frame is, the less precision the tasks lose.

Existing privacy-preserving systems for video analytics fall into two categories: detection-based ROI (region of interest) protection and global protection. The detection-based ROI protection [10], [13] employs a detection model to locate the ROIs and denature the ROIs with some common methods, e.g., face de-identification and inpainting. And the global protection [8], [18], [19] applies an effect to the whole video frame without content-aware techniques, e.g., GAN-based cartoonlization, image filtering. The techniques adopted by these systems are numerous and complicated, including traditional computer vision (e.g., OpenCV library), convolution neural network (CNN), GAN, etc. These techniques show their special advantages in different video analytics tasks.

B. Motivation for X-Stream

a) Application-specific privacy protection: Surveillance cameras are widely deployed in our daily life, ranging from streets to rooms. The analytics tasks vary in different scenarios, for example, deploying cameras in elder people's houses aims to alarm at the first time they fall; cameras in the street are used to detect illegally parked vehicles. These analytics tasks

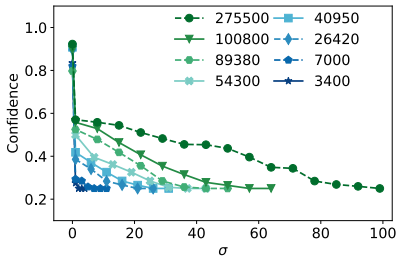


Fig. 1. The detection confidence variation under different σ.

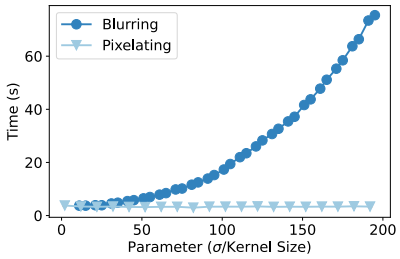


Fig. 2. The runtime of the blurring and pixelating.

need different information from videos, and at the same time, they are required to protect different privacy information.

Application-specific privacy protection is often cumbersome due to varying privacy-preserving targets from scene to scene. For instance, in outdoor scenarios, individuals may be reluctant to expose their license plates, a potential identity reveal. The conventional approach is manual blurring of the target area. However, in indoor settings like private rooms, concerns may shift to decor and sensitive personal effects, rendering manual methods inefficient. Unfortunately, there's no one-size-fits-all solution to fulfill diverse privacy protection needs, requiring developers to delve into video processing libraries and different privacy protection methods. Our aim is to offer developers or users seeking privacy protection for video analytics tasks a flexible API that can safeguard and expose information based on their specific requirements.

b) The impact of dynamic video content on privacy protection: Besides the application-specific requirement, the video source in VSA is also dynamic. The dynamic video content brings a new challenge to the privacy protection. Video content varies from frame to frame, which affects the efficiency of privacy protection and even makes some techniques lose efficacy. We take Gaussian blur [20], a widely used privacy protection technique, as an example. The key parameter σ in Gaussian blur* controls the sharpness: the higher σ is, the more blurry the frame is, and hence the less information it contains. We apply Gaussian blur in different σ to frames that show objects with different sizes, and detect the object in the blurry frames. Figure 1 illustrates the confidence of object detection under different object sizes as σ increases. We find that when σ reaches 40, the small object is undetectable while the large object can be detected easily with high confidence. If we all adopt the highest parameters to protect the objects, in the extreme blacken the objects out, we may lose the intelligibility of the objects. For example, we just want to protect human faces, but the overwhelming blurring takes the information of person away, and thus the person becomes unrecognizable. There are some other techniques whose utility is affected by parameters, such as mosaic and cartoonlization [21]. The results motivate that the privacy protection techniques should be fine-tuned according to the dynamic video content.

Additionally, the advantage of dynamically changing key parameters and switching protection methods is that the system can adapt to changeable and complex video content and perfectly protect privacy at any time. Moreover, in scenarios with low privacy exposure risk, adopting lighter privacy protection methods can significantly conserve computing resources, enhancing efficiency. As shown in Figure 2, when the blurring parameter is low, the overhead is comparable to pixelating. However, with a high parameter, the overhead is unacceptable for the real-time privacy protection requirement. Given that blurring preserves more information than pixelating, the preference is for blurring when the overhead is manageable. This necessitates method switching to mitigate system overhead, a feature overlooked in previous works.

III. System Overview

A. Overview

The X-Stream architecture, depicted in Figure 3, comprises four key components: **PriQL Parser**, **Content Analyzer**, **Methods Mapper**, and **Protector**. The PriQL Parser transforms the owner's PriQL sentence requirements (discussed in Sec. IV-A) into a dictionary format. Content Analyzer analyzes the video content according to the video features, and filters which video frames to be processed. Methods Mapper breaks users' requirements by the protection dimension (introduced in Section IV-C) and maps the dictionary-form requirement to a DAG of protection methods for each filtered frame, considering the dynamic video content. Protector renders video frames according to the DAG and produces the protected video. In case of simultaneous tasks, Protector merges common DAG parts to enhance processing speed.

The multi-task privacy-preserving video transformer works as follows. Consider a traffic video showing cars and pedestrians whose privacy should all be protected. There are two tasks in this video, i.e., pedestrian counting and car counting. To ensure intelligibility, the owner should expose the body outline and protect the face for the pedestrian counting task, and for the car counting task, the owner should expose the car outline and protect the license. In the beginning, based on the requirements, the owner inputs the PriQL sentences (as shown in Figure 5) to X-Stream. PriQL Parser parses the two PriQL sentences and outputs the dictionary-form requirements, including video sources, protection list, and expose list. The Content Analyzer captures the features from the source video, such as the pixel difference between consecutive frames, to help Methods Mapper decide the suitable protection

*To blur pixels in the frame, Gaussian blur filters each pixel using the weighted average of the group of pixels surrounding the target. The weight is calculated by Gaussian distribution and σ is the standard deviation which determines the effect of surrounding pixels on the target pixel's value.

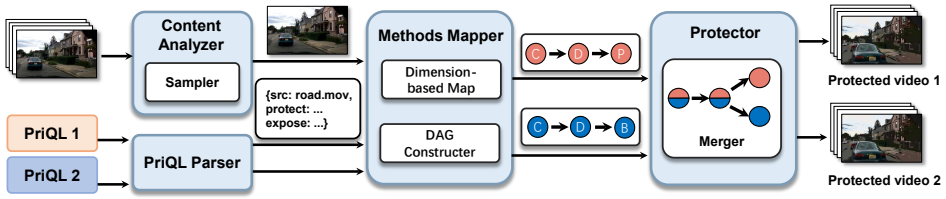


Fig. 3. System architecture of X-Stream.

```
SELECT *                              expr definition
FROM src_video_name                   expr :=   elem [and expr]
[WHERE [timestamp1, timestamp2]]      elem := dim='item'
PROTECT expr [, ...]                  dim := [object | motion | background]
[EXPOSE expr [,...]]                  item := [A  -Za-z0-9]+
```

Fig. 4. The syntax of PriQL.

```
SELECT *                              SELECT *
FROM crossroad.mp4                    FROM cafe.mp4
WHERE timestamp = [0, 20]             WHERE timestamp = [0, 20]
PROTECTobject = 'license plate'       PROTECT object = 'face'
EXPOSE object = 'car'                 EXPOSE object = 'person'
```

Fig. 5. Examples of PriQL.

method with appropriate parameters. To speed up the video stream transforming, Content Analyzer also decides which video frames should be processed. Then, Methods Mapper constructs one processing DAG for each filtered frame for each requirement and merges all DAGs to improve time efficiency. As shown in Figure 3, for Frame 1, DAG of PriQL 1 contains cartooning (C), detection (D), and pixelating (P), while DAG of PriQL 2 consists of cartooning (C), detection (D), and blurring (B). Protector produces two protected videos according to the merged DAG where the common methods, i.e., cartooning and detection, are only processed once. By sharing the intermediate result in DAGs, the transforming cost for the multi-task requirements can be reduced significantly.

IV. Key Design

A. PriQL

To address the tedious and complicated privacy protection issue, we introduce PriQL, an SQL-like language for generating the privacy-preserving video stream. Video owners who are familiar with SQL can easily generate privacy-enhanced videos by writing familiar PriQL sentences. PriQL frees video owners from learning tedious and obscure computer vision libraries and complicated privacy protection methods. They only need to focus on the requirement of privacy protection and intelligibility of video streams.

a) PriQL language syntax: Figure 4 shows the PriQL syntax. The object of `SELECT` is the video stream, and `WHERE` restricts the time interval of the video which will be shared to subscribers. PriQL supports owners' requirements of privacy protection and intelligibility by extending SQL with two keywords: `PROTECT` and `EXPOSE`, where `PROTECT` allows owners to describe their requirements of privacy protection, and `EXPOSE` allows owners to point out the preserved information for VSA tasks. The detail of *expr* in the `PROTECT` and `EXPOSE` syntax is shown in Figure 4. We next provide the motivation behind each additional piece of syntax.

First, the `PROTECT` keyword allows owners to customize the privacy-protected content, while the `EXPOSE` keyword ensures the video frame is still eligible for the VSA tasks after applying the protection. If PriQL is without `EXPOSE` keywords, we can use encryption to protect all the information in the video stream and achieve the protection goal. However, the protected video is unqualified for VSA tasks due to the lack of intelligibility.

Second, there is a natural trade-off between protecting privacy and providing intelligibility, as the former seeks to conceal information in the video while the latter requires exposing certain information for VSA. Section IV-D will discuss the parameter fine-tuning approach to improving the intelligibility while ensuring privacy protection.

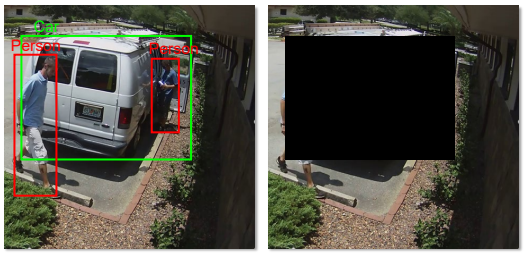
Third, according to our observations and the settings in the state-of-the-art, there are three key dimensions of privacy information, i.e., the object in the scene, motion information, and background. To endow PriQL with the ability to describe these privacy dimensions, we design the *expr* with three-dim words: object, motion, and background, as depicted in Figure 4. Using these words and their combinations, we can express multifarious protection requirements. For example, we can write `object='person', background='True'` to protect the indoor decoration and person identity. And if we want to protect the motion information, we can use `motion='True'`.

Lastly, owners are able to describe their exposure requirement by `EXPOSE` keyword. The format basically forms the same as `PROTECT` keyword. If owners want to send the video to do the car counting task, they can write `EXPOSE object='car'` to inform our system that cars in the scene need to be exposed. Note that when there are conflicts between protection and exposure, we prefer to protect rather than expose since our system's primary objective is privacy-preserving. As to how to deal with the preference, we will introduce it in the next subsection.

b) PriQL sentence examples: Figure 5 shows two concrete examples to explain how the requirements can be written in PriQL The details are given in Section IV-A.

B. Content Analyzer

The video content is crucial for privacy protection, as discussed in Section II-B. To take advantage of the video content, we design a content analyzer that is dedicated to extracting the key features from video frames. Specifically, Context Analyzer should capture the motion-related features, such as pixel difference between consecutive frames [22], to



(a) Detection before protection (b) Detection after protection

Fig. 6. Protection interferes the detection result.

estimate objects' speed of motion and help Methods Mapper to employ motion protection. Moreover, the pixel size of objects also needs to be extracted for fine-tuning parameters of some protection methods, as discussed in Section IV-D in detail.

C. Methods Mapper

As mentioned in Section IV-A, we design PriQL to shed light on three dimensions of privacy protection. To cooperate with PriQL and enable the protection, we design a powerful content-aware Methods Mapper, which exploits owners' privacy requirements described in PriQL sentence and content of video frames to generate an efficient processing DAG with video operators (i.e., single protection method).

a) Protection dimensions: As discussed in Section IV-A, the key dimensions of privacy protection are objects in the scene, motion information, and background. These dimensions are orthogonal for privacy protection, making it hard to use a single protection method to conceal two of them. For example, detection-based methods are efficient for object protection, such as inpainting which detects ROIs and denatures them. However, these methods have poor performance on the background protection since the background (e.g., the indoor decoration) is hard to detect. Additionally, the detection-based method cannot handle gait detection, i.e., recognize persons' identification by their walk pattern. We thus need a new solution for protecting motion information. To this end, we propose a dimension-aware protection scheme. Thanks to PriQL, Methods Mapper can easily divide owners' requirements into different dimensions.

b) Protecting objects: Object protection requires the detection-based method, such as YOLOv5 [23] and Mask R-CNN [24]. Given a series of objects that need to be protected, we need to detect all the ROIs where objects are located.

After locating objects in the video frame, we need to denature them for privacy protection. Considering the class, size, and location of the object, different denaturing methods should be taken on different targets. X-Stream takes into account three common denaturing methods, i.e., blurring, pixelating (i.e., mosaic), and black-out (i.e., directly blackening the object). As discussed in Section II, the intelligibility of the frame varies with the parameter setting in these methods. Method Mappers determines the denaturing methods with fine-tuned parameters for each object, whose details will be discussed in the following Section IV-D.

c) Protecting background: We use cartooning, a representative filter-based method, to protect the background. Given a video frame, cartooning first applies the bilateral filter [25], an edge-preserving smoothing filter, to reduce color numbers. Cartooning then performs K-means algorithm [26] to cluster the histogram of the image and recolors the image with clustered pixels. Lastly, cartooning applies the contour finding algorithm [27] to enrich the video frame with more lines.

We adjust the number of clusters, which is denoted as k, in the K-means algorithm. In general, the more clusters K-means has, the more vivid the frame is, and the more privacy would be disclosed. By using the adaptive algorithm [26], Methods Mapper can choose the optimal k for K-means automatically.

d) Protecting motion: We find that filtering frames from the video stream can make people's motion discontinuous, which provides the opportunity for motion protection. It is noted that frame filtering has little influence on VSA tasks. That is because consecutive frames are relatively identical within a very short time interval in general. Consider the subscriber wants to count pedestrians from a surveillance video recorded in a square. If we sample the raw video from 30 FPS down to 5 FPS, only the perceptive quality of the video is degraded rather than the intelligibility of the people counting since the people movement can still be captured generally.

Methods Mapper utilizes the random sampling technique to protect motion information. More specifically, the technique adjusts the sampling rate according to the object's speed of motion and selects the random number of frames in a certain time interval. The speed of motion is measured by the frame difference [28] using the OpenCV library.

e) Exposure: PriQL provides owners an interface to describe objects and other dimension information which video owners want to expose just like the keyword `PROTECT`. Since the `EXPOSE` keyword is identical with `PROTECT`, the scheme behind the exposure is similar to the protection. Specifically, we should first detect the objects that need to be exposed, and then try to preserve them in video frames. However, object protection and exposure are always tangled. Protecting one object may sometimes have an effect on the intelligibility of other objects that need to be exposed. As shown in Figure 6, two persons in the original video frame can be well detected, while the persons vanished when we use the black-out method to protect cars in the scene. To balance the tradeoff between protecting privacy and preserving intelligibility, we propose two empirical approaches as follows.

First, as a privacy-enhancement system, X-Stream should provide privacy protection as the top priority. That is, when video owners require to protect certain objects, X-Stream must guarantee these objects will not be revealed. To preserve the intelligibility of the video analytics task, the protection method needs to have a limited effect on other objects related to the analytics. To this end, we design a parameters tuning approach to maximize the intelligibility of objects that owners need to expose, which will be demonstrated in the next Section IV-D.

Second, objects need to be protected and exposed sometimes are overlapping in some scenarios (as shown in Figure 6).

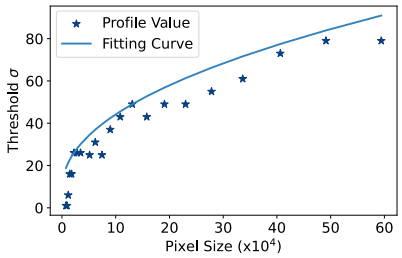


Fig. 7. The relationship between the pixel size and threshold σ.

In this case, Methods Mapper will adopt a mild protection method, like cartooning, to the object that needs to be exposed, which guarantees privacy at the expense of limited degradation of intelligibility.

D. Parameter Tuning

To provide object protection with blurring at the highest quality, we first capture the threshold of σ under different pixel sizes of objects, as shown in Figure 7. The mathematical relationship estimated by the curve fitting is illustrated by the green line in Figure 7. The expression of the fitting curve is given by $\hat{\sigma} = 0.1694 \times p^{0.4658}$, where p indicates the pixel size of the object, σ is the estimated minimum parameter in the blurring satisfying the requirement of privacy protection. To further ensure privacy protection, we need to introduce a bias θ that is more appropriate in this case to the fitting curve and present the threshold of σ as $T_\sigma = \hat{\sigma} + \theta$.

We note that it is impractical and inefficient to aggressively increase σ. As discussed in Section II, the processing time of the blurring increases as σ. Although the blurring has little effect on the intelligibility and owns comparable privacy-preserving performance, its overhead is still excessive when the parameter σ gets higher. To balance the tradeoff between time efficiency and privacy-preserving performance, Methods Mapper will switch to pixelating for object protection if the estimated threshold of σ exceeds 90.

Different protection methods show distinct system overhead. We use blurring, pixelating, and cartooning, respectively, to process a 40-second video stream with 30 FPS. Figure 2 compares the total processing time of the blurring and pixelating. The minimum time consumption of cartooning is around 0.45 seconds per frame, which is much slower than the other two methods, so we omit this result in the figure. The results show that blurring is more time-consuming than pixelating when its parameter σ gets higher (i.e., the protection gets stronger).

V. Optimization

A. Inter-frame Inference and Tracking

As discussed in Section IV-C, object protection and exposure should first detect objects that need to be concealed or exposed. However, existing object detection models still cannot provide the optimal accuracy over a large number of object classes [29]. Inevitably, these objects will be exposed or concealed accidentally in some video frames if the object detection model fails to detect them.

We present the inter-frame inference to avoid privacy exposure or intelligibility degradation due to the detection failure. For the video stream, the object location in consecutive frames will not change significantly in a short time (e.g., within 100 ms) [30]. Hence, the inter-frame inference can estimate the object's location in the current frame using the detection results in prior frames. To this end, we combine the detector with the object tracking approach [31]. More specifically, we search the bounding boxes in the frame $i+1$ (denoted as candidate object) locate around the box of the target object in the frame i. If there is a candidate object satisfying that the overlapped area of the candidate and the target object (in terms of the intersection over union) or their distance are within thresholds, we consider that the target object in the frame $i+1$ is detected. Otherwise, the target object is lost by the detector and the object tracker will be issued to estimate the bounding box in the frame $i+1$. The estimation result will also be recorded for the next inter-frame inference process.

For the object tracker, it is non-trivial to judge whether a target object is lost by the detector or just moves out of the scene. To this end, we utilize the relative motion direction and the relative location of the target object to help track the object efficiently. Specifically, we derive the relative motion direction by calculating the distance between the centroids of bounding boxes in consecutive frames. As the object near the frame border is more likely to move out of the scene, the tracker should mainly focus on their relative motion direction. Take a target object located in the rightmost corner as an example. If the detector fails to capture it and the relative motion direction is right, we then assume this object is away from the scene and will not issue the tracker. Otherwise, if the relative motion direction is opposite, the tracker should be issued to conceal privacy or preserve intelligibility. In this case, if the tracker also fails to track the target, this object is regarded as moving away.

B. Spatial and Temporal Correlations

X-Stream incorporates the spatial correlation between two users' requirements to reduce the system overhead, as discussed in Section III. In general, if two or more requirements over the same video stream are specified X-stream, we first generate DAGs for each requirement and identify the common parts (i.e., video operators) of the DAGs. Then the protector merges the DAGs and processes the common cells just once time for saving time. The distinct parts are processed as usual. The resource utilization and time efficiency can be further improved by dynamic resource configuration [32], [33], DNN partitioning [34], [35], parallel processing [36], [37], workload management [38], transfer learning [39], etc., which we leave to future work.

From our observation, the frames of video streams souring from surveillance cameras have a strong spatial correlation since the frames within a short time are identical. Exploiting this feature, we present a sampling technique in Section IV-C to protect the motion information, which can also be used to reduce the system overhead. Considering a 40-second video

with 30 frames per second (FPS), if we take every single frame as the key frame and transform it, the processing time is about 480 seconds running on our machine (configuration is shown in Section VI. There are a variety of redundant frames because of the high similarity of the consecutive frames. Thus, we sample the video based on the motion in frames. When the motion is fast in the frame, the sampling rate is high. If the motion is slow in the frame, the sampling rate decreases. When we apply the sampling technique to the video, the processing time reduces to 16 seconds as shown in Figure 9(b).

VI. Implementation

a) Hardware: We deploy our system X-Stream on a high-end machine with two 26-core Intel® Xeon® Platinum 8269CY CPUs, DRAMs of 190 GB in total, and an Nvidia® RTX 6000 GPU with 24 GB VRAM.

b) Software: We implement X-Stream in Python 3.7 and use PyTorch v1.9.1 for training and evaluating. We use YOLOv5 with weights YOLOv5s as the backbone detection model, and adopt the facenet [40] as the face detection model. Since the YOLOv5 cannot label the license plate, we further train our license plate detection model using YOLOv5 and the dataset from Kaggle [41]. The model achieves comparable accuracy with the pretrained YOLO model.

We use OpenCV to process video frames. We use the regular expression to parse the PriQL queries in Python, and save the query elements into a dictionary for further processing. We employ cartooning [21] to protect the background information, and use the random-sampling technique to protect the motion information. Blurring and pixelating for protecting objects are implemented with OpenCV.

X-Stream presents video owners with an API in either command line format or embedded format. The command line format API allows owners to transform video streams into protected videos that fulfill their privacy requirements by running the application we present on Linux. The embedded format allows owners to embed our function into their own applications using the PirQL sentence when they are developing applications.

VII. Evaluation

A. Datasets

We evaluate the system using video streams from the following two sources:

- BDD - the Berkeley DeepDrive dataset [42] consists of 1,100 hours of videos recorded by dashboard cameras on motor vehicles at 30 FPS. Each video spans 40 seconds with HD (1280×720) resolution, including day and night scenes from urban to rural.
- YOUTUBE - HD (1280×720) video streams from YouTube which are shot by surveillance cameras on a campus, cafe, or street at 30 FPS.

B. Baselines

We compare our solution with four baseline approaches and their enhanced variants for video transformation:

a) Cartoon: We incorporate the PECAM [8] as one of our baselines. PECAM employs a GAN to generate the cartoonlized video frames to protect privacy, i.e., it takes the cartoon effect to the whole video frame. For simplicity, we use the OpenCV to cartoonlize the whole video frame which has the same effect as the PECAM without any detection model. The cartoonlized method is the same as the methods we use to protect background information in X-Stream, and the detail of the implementation can be found in Section IV-C.

b) Blurring: Technically, the method uses the detection model (e.g., YOLOv5) to locate privacy-sensitive objects that reside in the pre-defined list, and then overlap each privacy-sensitive object with an irrelevant avatar.

c) Pixelating: Pixelating (a.k.a. mosaic) is a frequently-used privacy-preserving method. As manually pixelating privacy-sensitive objects is tedious and cumbersome, we employ a detection model to help detect the privacy-sensitive objects and then we pixelate the areas where the objects reside with a 6×6 pixel kernel. The value of a pixel is calculated using weighted mean from the surrounding pixel blocks in the size of 6×6.

d) Blackout: For this method, there exists an inevitable tradeoff between privacy protection and intelligibility preservation, since the privacy-sensitive objects can only be totally blackened. During the experiments, we just blacken the area where the user-defined privacy-sensitive objects exist.

C. VSA Tasks and Metrics

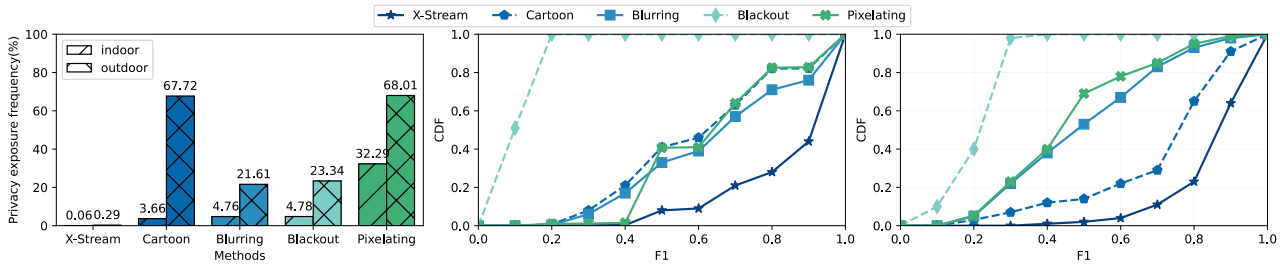
a) VSA tasks: To illustrate that X-Stream achieves both privacy protection and intelligibility preservation of video streams, the evaluation considers two common VSA tasks, i.e., object detection and background extraction.

We categorize videos from the dataset into indoor and outdoor scenarios, tailoring VSA tasks based on owners' preferences. Outdoor videos typically capture intersections, crossroads, and streets. In this setting, the video owner prioritizes protecting license plates while exposing cars for counting. The PriQL sentence appears in the left part of Figure 5. Indoor scenarios feature locations like restaurants, banks, salons, and pharmacies. Here, the owner conceals faces and exposes body figures for people counting. The PriQL sentence is found in the right part of Figure 5. Additionally, we address background extraction tasks [43] for both scenarios, using the same PriQL sentences as for object detection.

b) Metrics: We utilize the license recognition algorithm [44] and face detection model [40] to assess privacy enhancement in outdoor and indoor scenarios, respectively. The exposure frequency, defined as the ratio of detected licenses (faces) to the total licenses (faces) in the video stream, serves as a metric for privacy performance. A lower exposure frequency indicates better privacy enhancement. The F1 score is employed to evaluate VSA task performance and quantify the preserved intelligibility of each approach.

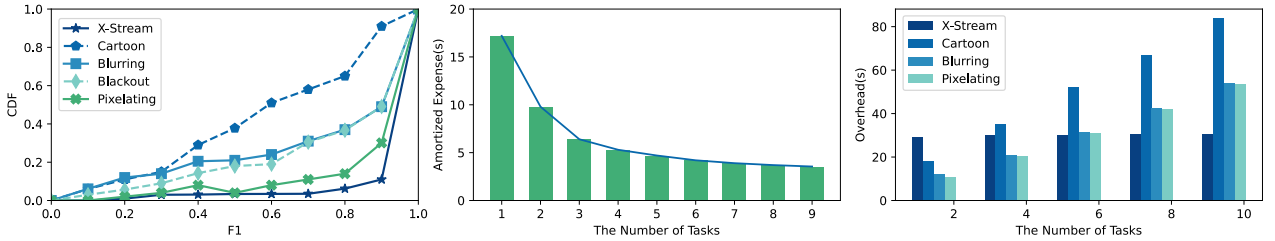
D. Privacy Protection and Intelligibility Preservation

a) Privacy loss: Privacy loss depends on the number of wrongly exposed privacy-sensitive objects that we care



(a) Privacy Preserving. (b) F1 score of the car detection task. (c) F1 score of the human detection task.

Fig. 8. Privacy protection and intelligibility preserving under different video scenarios.



(a) F1 score of the background extraction task. (b) Multi-tasks amortized expense. (c) Multi-tasks runtime under different methods.

Fig. 9. The performance under the background extraction task and X-Stream system overhead.

about. For outdoor scenarios, video owners would like to protect license plates when transforming their traffic surveillance videos. Figure 8(a) compares the exposure frequency of license plates in the car counting task over X-Stream and baseline approaches. The results show that X-Stream yields the best performance as compared to all baselines. Specifically, compared with the cartoon and pixelating, X-Stream achieves up to 200× fewer exposure times over five video streams. For the indoor scenario, the privacy-sensitive object switches to the human face as declared in the PriQL sentence in Figure 5. As illustrated in Figure 8(a), X-Stream just exposes 0.06% of the total 8660 faces in five indoor videos.

We also find that the cartoon shows poor performance especially when protecting license plates in the outdoor scenario, since the cartoon is hard to deal with the case that the privacy-sensitive object covers a large area of the scene. When the object size is large, the protected license plates and human faces can be recognized, and hence the cartoon is invalid. X-Stream addresses this issue by dynamically tuning parameters. The blurring is also not ready for the privacy-preserving requirement since it highly relies on the performance of the object detection model. The failure of object detection will lead to the exposure of privacy-sensitive objects. X-Stream alleviates this problem by introducing the inter-frame inference approach, which will be illustrated in Section VII-F in detail. In short, X-Stream outperforms all baselines.

b) Intelligibility preserving: We first focus on the performance of intelligibility preserving for the object detection tasks. Figure 8(a) and Figure 8(c) show the F1 score achieved by the car counting task in the outdoor scenario and the human counting task in the indoor scenario, respectively. The results show that the protected videos transformed by X-Stream achieve the best F1 score. The preservation of intelligibility is derived from two features that have been introduced in our system. The first one is that we take the users' exposure requirement into account, and when the exposure areas have no effect on the privacy loss, we take the specific area of the raw frame to replace the protected one. The second feature is our fine-tuning parameter techniques. The lowest parameters of methods can preserve the intelligibility of frames as much as possible from the experimental results. We also note that the blackout performs the worse among X-Stream and all baseline approaches. That is because the blackout just blackens the whole car and person out for privacy protection, and leaves no information for detection.

We also investigate the intelligibility preservation for the background extraction tasks which are used to remove the background of video frames for further processing such as activity recognition [45]. Figure 9(a) depicts the F1 score for background subtraction in the outdoor scenarios. We observe that X-Stream achieves the best intelligibility preservation. It is worth noting that the pixelating method has relatively similar performance to X-Stream, which however, is at the expense of high privacy loss, as illustrated in Figure 8(a).

E. System Overhead

X-Stream leverages the multi-task merging to speed up the video transforming. We evaluate the system overhead of X-Streamunder multiple PriQL requirements on one video stream. We consider a group of tasks that all need to protect the background and license plates in a traffic video that spans 40 seconds with 30 FPS. Figure 9(b) shows the amortized time cost of transforming videos for multiple tasks. We find that when there is only one PriQL requirement, it takes about 17.5 seconds for video transforming. When there are 5 PriQL requirements for this video stream, the amortized time reduces to around 5 seconds, i.e., the amortized system overhead decreases by more than 2×. More PriQL requirements are

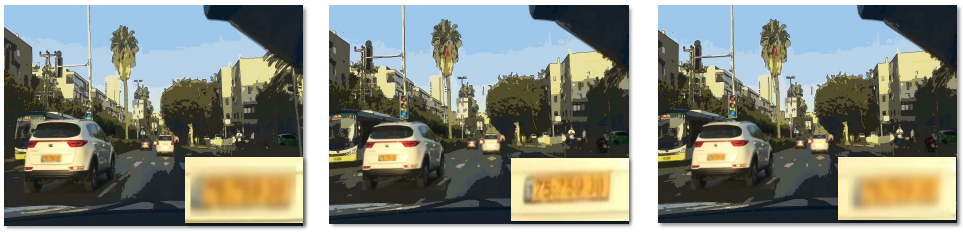


Fig. 10. An example of two consecutive video frames under a detection failure.

involved, and less amortized expense will be incurred. When there are 9 concurrent PriQL requirements, the amortized time is close to 2.5s, whose speed-up rate is 7×. It is also worth noting that the time cost of handling one PriQL requirement is less than the video's time duration, indicating the great time efficiency and thus the practicality of X-Stream.

We further compare the system overhead among X-Stream and all baseline approaches. Here, X-Stream transforms 5 videos from the BDD dataset considering the same PriQL requirements. The averaging transforming time is shown in Figure 9(c). When there is only one PriQL requirement, X-Stream takes more time to process videos since it utilizes several methods for privacy protection and intelligibility preservation. As the number of concurrent VSA tasks increases, X-Stream only brings limited marginal overhead while the overhead of baselines grows proportionally. In short, by taking advantage of spatial and temporal redundancy, X-Stream achieves a great time efficiency compared with baseline approaches.

We profile the memory expense when running multiple tasks. The results show that even when the 9 tasks run at the same time, the load and store memory expense is no more than 10 MB, which is lightweight and can be deployed in practice.

F. Gain from Tracking

X-Stream leverages object detection to protect privacy-sensitive objects and preserve intelligibility. As discussed in Section V-A, the failure of object detection will lead to privacy leakage. To this end, we design the inter-frame inference and tracking to ensure the efficiency of X-Stream when the object detection fails. Figure 10 illustrates an example of two consecutive video frames under detection failure. We can find that the license plate in the frame i is protected since the object detection model works well. However, the frame $i+1$ may expose the identity of this car because of the exposed license plate. When utilizing the inter-frame inference and tracking technique, the location of the license plate in the frame $i+1$ can be tracked accurately although the detection fails, avoiding the privacy issue efficiently.

VIII. Related Work

A variety of works have been proposed, whose techniques are diverse from each other [8], [13], [15]. PECAM takes a GAN-based cartoon transformer as the privacy-preserving method [8]. Hassan et al. use CNNs to detect the sensitive objects they pre-defined and cover the objects with cartoon avatars [13]. Li et al. combine CNN and GAN to generate fake human faces, which are used to obfuscate human beings and neural network detection models [15]. The techniques behind these systems are overwhelming and complicated. Unlike prior works, X-Stream provides video owners an API to describe their privacy and exposure needs and leave the annoying protection methods to our powerful back-end of the system. We also take the dynamical parameters tuning into account, which none of the prior works have considered.

Recent work has also focused on obfuscating video streams against computer vision rather than human beings to reduce privacy leakage [18], [19]. Fawkes [19] trains an obfuscation mechanism, which injects the original video frame into the training process. However, the training process is time-consuming and compute-intensive, which hurts the usability of the system. This obfuscated video may also be unsuitable for human viewing. Liu et al. transform the video frames to a special form, which cannot be understood by humans, while is comprehensible for the neural network [18]. This method also cannot support the video review. X-Stream works with off-the-shelf object detectors and provides human-viewable video streams that are usable by unmodified applications.

Declarative query languages are commonly used in video data query systems [9], [46], [47], which provide a versatile interface for users to query video data. Kang et al. present a SQL-like language to query video segments [46]. However, the target is a video database, not live video streams as we consider in this paper. With Privid [9], users can use the language the authors design to obtain the results in text form from the target video. X-Stream incorporates the advantages of declarative query languages, and extends the SQL with a set of new keywords to fulfill our design requirements.

IX. Conclusion

X-Stream is a video stream analytics privacy enhancement system with a declarative query language, PriQL. The PriQL provides video owners an interface to describe their privacy protection and exposure requirements. X-Stream parses the PriQL sentence and produces a processing DAG to transform the video streams. X-Stream also dynamically adapts the DAG based on the real-time video context information. The output of X-Stream can be directly taken as inputs by various existing VSA tasks with privacy protection. To further improve the performance, we present two key techniques, i.e., multi-task combination and inter-frame inference. Our results confirm that X-Stream can preserve the users' privacy and the intelligibility of VSA tasks, while being easy to use.

REFERENCES

[1] "Surveillance cameras are everywhere. and they're only going to get more ubiquitous." 2020, https://crimereads.com/surveillance-cameras-are-everywhere-and-theyre-only-going-to-get-more-ubiquitous/.

[2] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *USENIX NSDI 2017*, 2017, pp. 377–392.

[3] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *IEEE INFOCOM 2018*. IEEE, 2018, pp. 1421–1429.

[4] V. Nigade, P. Bauszat, H. E. Bal, and L. Wang, "Jellyfish: Timely inference serving for dynamic edge networks," in *IEEE RTSS 2022, Houston, TX, USA, Dec. 5-8, 2022*. IEEE, 2022, pp. 277–290.

[5] V. Nigade, L. Wang, and H. E. Bal, "Clownfish: Edge and cloud symbiosis for video stream analytics," in *IEEE/ACM SEC 2020, San Jose, CA, USA, November 12-14, 2020*. IEEE, 2020, pp. 55–69.

[6] "EU makes move to ban use of facial recognition systems," 2020, https://sciencebusiness.net/news/eu-makes-move-ban-use-facial-recognition-systems.

[7] "San francisco is first us city to ban facial recognition," 2019, https://www.bbc.com/news/technology-48276660.

[8] H. Wu, X. Tian, M. Li, Y. Liu, G. Ananthanarayanan, F. Xu, and S. Zhong, "PECAM: privacy-enhanced video streaming and analytics via securely-reversible transformation," in *ACM MobiCom 2021, New Orleans, Louisiana, USA, October 25-29, 2021*. ACM, 2021, pp. 229–241.

[9] F. Cangialosi, N. Agarwal, V. Arun, J. Jiang, S. Narayana, A. Sarwate, and R. Netravali, "Privid: Practical, privacy-preserving video analytics queries," *CoRR*, vol. abs/2106.12083, 2021.

[10] O. Gafni, L. Wolf, and Y. Taigman, "Live face de-identification in video," in *IEEE/CVF ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 9377–9386.

[11] H. Yu, J. Lim, K. Kim, and S. Lee, "Pinto: Enabling video privacy for commodity iot cameras," in *ACM SIGSAC CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. ACM, 2018, pp. 1089–1101.

[12] M. Ra, R. Govindan, and A. Ortega, "P3: toward privacy-preserving photo sharing," in *USENIX NSDI 2013, Lombard, IL, USA, April 2-5, 2013*. USENIX Association, 2013, pp. 515–528.

[13] R. Hasan, P. Shaffer, D. Crandall, E. T. Apu Kapadia *et al.*, "Cartooning for enhanced privacy in lifelogging and streaming videos," in *IEEE/CVF CVPRW 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 29–38.

[14] Q. Sun, L. Ma, S. J. Oh, L. V. Gool, B. Schiele, and M. Fritz, "Natural and effective obfuscation by head inpainting," in *IEEE CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE, 2018, pp. 5050–5059.

[15] T. Li and L. Lin, "Anonymousnet: Natural face de-identification with measurable privacy," in *IEEE CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 56–65.

[16] M. Maximov, I. Elezi, and L. Leal-Taixé, "CIAGAN: conditional identity anonymization generative adversarial networks," in *IEEE/CVF CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 5446–5455.

[17] "dlib c++ library," 2019, http://dlib.net/.

[18] S. Liu, J. Du, A. Shrivastava, and L. Zhong, "Privacy adversarial network: representation learning for mobile data privacy," *ACM IMWUT*, vol. 3, no. 4, pp. 1–18, 2019.

[19] S. Shan, E. Wenger, J. Zhang, H. Li, H. Zheng, and B. Y. Zhao, "Fawkes: Protecting privacy against unauthorized deep learning models," in *USENIX Security 2020, August 12-14, 2020*. USENIX Association, 2020, pp. 1589–1604.

[20] "Gaussian blur," 2021, https://en.wikipedia.org/wiki/Gaussian_blur.

[21] "Cartoonizer," 2021, https://github.com/lutianming/cartoonizer.

[22] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *ACM SIGCOMM 2020, Virtual Event, USA, August 10-14, 2020*. ACM, 2020, pp. 359–376.

[23] "yolov5," 2021, https://github.com/ultralytics/yolov5.

[24] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *IEEE ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE, 2017, pp. 2980–2988.

[25] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *IEEE ICCV 1998, Bombay, India, January 4-7, 1998*. IEEE, 1998, pp. 839–846.

[26] G. Hamerly and C. Elkan, "Learning the k in k-means," *Advances in neural information processing systems*, vol. 16, pp. 281–288, 2004.

[27] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[28] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *ACM SenSys 2015*, 2015, pp. 155–168.

[29] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo, "Swin transformer V2: scaling up capacity and resolution," in *IEEE/CVF CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 2022, pp. 11 999–12 009.

[30] B. Feng, Y. Wang, G. Li, Y. Xie, and Y. Ding, "Palleon: A runtime system for efficient video processing toward dynamic class skew," in *USENIX ATC 2021*, 2021, pp. 427–441.

[31] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[32] Z. Wen, Y. Wang, and F. Liu, "Stepconf: Slo-aware dynamic resource configuration for serverless function workflows," in *IEEE INFOCOM 2022, Virtual conference, May 2-5, 2022*. IEEE, 2022, pp. 1868–1877.

[33] F. Xu, J. Xu, J. Chen, L. Chen, R. Shang, Z. Zhou, and F. Liu, "igniter: Interference-aware GPU resource provisioning for predictable DNN inference in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 812–827, 2023.

[34] J. Wu, L. Wang, Q. Jin, and F. Liu, "Graft: Efficient inference serving for hybrid deep learning with slo guarantees via dnn re-alignment," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–18, 2023.

[35] J. Wu, L. Wang, Q. Pei, X. Cui, F. Liu, and T. Yang, "Hitdl: High-throughput deep learning inference at the hybrid mobile edge," *IEEE Trans. Parallel Distributed Syst.*, vol. 33, no. 10, pp. 4499–4514, 2022.

[36] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang, "Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading," in *ACM MobiCom 2021, New Orleans, Louisiana, USA, October 25-29, 2021*. ACM, 2021, pp. 201–214.

[37] A. Chen, F. Xu, Y. Dong, L. Chen, Z. Zhou, and F. Liu, "Opara: Exploring operator parallelism for expediting dnn inference on gpus," in *CCF CCFSys, Nanchang, China, August 4-5 2023*. CCF, 2023.

[38] Q. Pei, Y. Yuan, H. Hu, Q. Chen, and F. Liu, "Asyfunc: A high-performance and resource-efficient serverless inference system via asymmetric functions," in *ACM SoCC 2023, Santa Cruz, CA, USA, 30 October 2023 - 1 November 2023*. ACM, 2023, pp. 324–340.

[39] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "On-edge multi-task transfer learning: Model and practice with data-driven task allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1357–1371, 2020.

[40] "facenet-pytorch," 2021, https://github.com/timesler/facenet-pytorch.

[41] "Kaggle Dataset: Car License Plate Detection," 2021, https://www.kaggle.com/andrewmvd/car-plate-detection.

[42] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *IEEE CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE, 2017, pp. 3530–3538.

[43] O. Barnich and M. Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *IEEE Transactions on Image processing*, vol. 20, no. 6, pp. 1709–1724, 2010.

[44] "baidu open ai: ocr cars for plate," 2021, https://ai.baidu.com/tech/ocr_cars/plate.

[45] B. Garcia-Garcia, T. Bouwmans, and A. J. R. Silva, "Background subtraction in real applications: Challenges, current models and future directions," *Computer Science Review*, vol. 35, p. 100204, 2020.

[46] D. Kang, P. Bailis, and M. Zaharia, "Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics," *Proc. VLDB Endow.*, vol. 13, no. 4, pp. 533–546, 2019.

[47] F. Bastani, S. He, A. Balasingam, K. Gopalakrishnan, M. Alizadeh, H. Balakrishnan, M. J. Cafarella, T. Kraska, and S. Madden, "MIRIS: fast object track queries in video," in *ACM SIGMOD 2020, Portland, OR, USA, June 14-19, 2020*. ACM, 2020, pp. 1907–1921.