applied sciences

MDPI

Article

Privacy-Preserving Live Video Analytics for Drones via Edge Computing

Piyush Nagasubramaniam [1], Chen Wu [1], Yuanyi Sun [2], Neeraj Karamchandani [1], Sencun Zhu [1,*] and Yongzhong He [3]

[1] Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, USA; pvn5119@psu.edu (P.N.); cvw5218@psu.edu (C.W.); njk5270@psu.edu (N.K.)
[2] ByteDance Inc., Beijing 100098, China; yyspsu@gmail.com
[3] School of Computer, Beijing Jiaotong University, Beijing 100044, China; yzhhe@bjtu.edu.cn
* Correspondence: sxz16@psu.edu

Abstract: The use of lightweight drones has surged in recent years across both personal and commercial applications, necessitating the ability to conduct live video analytics on drones with limited computational resources. While edge computing offers a solution to the throughput bottleneck, it also opens the door to potential privacy invasions by exposing sensitive visual data to risks. In this work, we present a lightweight, privacy-preserving framework designed for real-time video analytics. By integrating a novel split-model architecture tailored for distributed deep learning through edge computing, our approach strikes a balance between operational efficiency and privacy. We provide comprehensive evaluations on privacy, object detection, latency, bandwidth usage, and object-tracking performance for our proposed privacy-preserving model.

Keywords: privacy-preserving; visual privacy; drone video analytics; edge computing; object detection

Citation: Nagasubramaniam, P.; Wu, C.; Sun, Y.; Karamchandani, N.; Zhu, S.; He, Y. Privacy-Preserving Live Video Analytics for Drones via Edge Computing. *Appl. Sci.* **2024**, *14*, 10254. https://doi.org/10.3390/app142210254

Academic Editor: Jose María Alvarez Rodríguez

Received: 14 October 2024
Revised: 31 October 2024
Accepted: 5 November 2024
Published: 7 November 2024

1. Introduction

The availability of smaller, lighter, and cheaper hardware has created a unique opportunity for the personal or commercial adoption of drones (also called unmanned autonomous vehicles) that are controlled by remote operators or onboard computers. With the capability of safely and quickly gathering data and reaching otherwise inaccessible locations, they open a world of possibilities for innovative drone applications such as goods delivery, aerial photography, entertainment, geographic mapping, search and rescue, and law enforcement. Thus, today drones are quickly increasing in numbers and complexity. As of February 2024, 781,781 drones have been registered, and 377,447 remote pilots have been certified by the FAA [1].

One of the most compelling capabilities of drones lies in their ability to capture live video footage from vantage points. Live video analytics using drones has emerged as a critical tool for unlocking the full potential of aerial imagery. By leveraging advanced algorithms and machine learning techniques, drones can extract valuable insights in real-time, enabling timely decision-making and proactive interventions. Whether it is identifying security threats, monitoring environmental changes, or optimizing agricultural practices, the ability to analyze live drone footage offers unparalleled advantages in various domains.

However, the process of conducting video analytics poses its own set of challenges, particularly when it comes to handling large volumes of data in real-time. Traditional approaches often rely on centralized cloud infrastructure for processing, which introduces significant latency and bandwidth constraints, especially in remote or bandwidth-limited environments. This limitation underscores the need for edge computing solutions that can perform data analysis closer to the source—on the drone itself or on nearby edge devices. However, the adoption of edge computing for live video analytics raises concerns about *visual privacy*, particularly when on the ground sensitive entities such as bystanders

or private spaces/properties are recorded without their consent or awareness. In edge computing scenarios, where processing tasks are offloaded to local servers or cloudlets situated near the edge of the network, there is a risk that sensitive information (e.g., the face of a stranger) in the camera view may be exposed to unauthorized entities. This potential breach of privacy underscores the importance of implementing robust privacy protection mechanisms in edge computing environments.

Intuitive privacy protection methods, such as data encryption, do not address the visual privacy problem because they only provide secure communication between drones and cloudlets without preventing cloudlets from accessing the data. Indeed, for the purpose of live video analytics, the cloudlet must access the visual data. Clearly, a good balance is needed for providing data utility while protecting visual privacy.

In light of these considerations, in this work, we propose a privacy-preserving framework for drones centered around live video analytics via edge computing. The major contributions of this paper are as follows:

- A novel framework for secure object detection and tracking utilizing a distributed deep learning architecture. We propose a method that preserves privacy while maintaining practical latency for real-time analytics through careful model selection on drones and the assistance of cloudlets.
- A lightweight method to prevent the leakage of sensitive information through feature maps, employing a combination of noise injection and a feature map selection strategy, with minimal compromise in accuracy.
- Comprehensive evaluations with real-world datasets, which demonstrate the effectiveness of visual privacy protection against reconstruction attacks as well as the effectiveness of video analytics in terms of accuracy and latency.

The proposed framework is discussed within the context of object detection and tracking but can be extended to similar Computer Vision tasks relevant to drones without loss of generality.

2. Related Work

Privacy-preserving computer vision models can be broadly categorized into models that (i) extract features from encrypted data (ii) extract secure features from raw data through the use of secure protocols, etc.

Ref. [2] presents a method for performing object tracking and motion detection on compressed, encrypted videos that leverages cloud computing for fast video processing. In this approach, a feature (called DNRC) is directly extracted from the encrypted bitstream to preserve privacy. In our system, we extract features from unencrypted video footage but make the transmitted data non-invertible to prevent data leakage.

Similar to our original problem, Ref. [3] addresses the issue of secure image feature extraction in a cloud-computing scenario using a secure feature extraction algorithm based on SIFT. The algorithm extracts "descriptors" from encrypted images, which can only be processed by the data owner. While this is interesting in the context of secure video processing in cloud-computing systems, it does not apply to modern computer vision models based on DNNs.

Ref. [4] presents a scheme for privacy-preserving video surveillance based on the Chinese Remainder Theorem. In this method, each video frame is transformed into seemingly random images on which further processing can be done. However, the video processing techniques are limited to traditional methods such as background subtraction and are not trainable like object detection neural networks.

In [5], a secure neural network inference framework called Gazelle is presented and it utilizes a combination of homomorphic encryption and garbled circuits to deliver secure inference. The focus of this work is on secure two-party communication, ensuring that users do not learn about the model while sensitive user data are protected from leakage. Our work is similar in that it provides secure inference, but our approach does not rely on encryption techniques. Instead, we leverage the inherent information loss in neural

network processing to deliver secure inference. Lastly, while Gazelle is a low-latency framework, it was not designed for use on edge devices with limited computing power.

Ref. [6] discussed an end-to-end cloud-based security framework called "DroneNet-Sec". The authors considered drones used for video analytics where a cloud and edge infrastructure supports the drones. However, their trust model includes the cloud infrastructure and the threats include man-in-the-middle attacks, denial of service, replay attacks, etc. We assume the cloudlet to be a semi-honest entity and, thus, adopt a fundamentally different approach to privacy protection by focusing on the information transmitted through the computer vision pipeline.

Ref. [7] presented a secure variant of the FasterRCNN object detection model with a focus on medical images. The framework also uses edge centers to minimize result latency. A key difference is that SecRCNN uses operations such as secure ReLU, secure convolution, etc., to perform secure feature map extraction. In our system, we refrain from modifying the fundamental operations on which the building blocks of the neural network are built in order to make the framework applicable to a variety of models. While our system works best with fast object detection models such as SSD-MobileNetV3, the system architecture can be extended to other models without loss of generality.

3. Preliminaries

In this section, we discuss the system model, security model, our design goals, and the challenges associated with designing a viable solution.

3.1. System and Security Models

We assume that the drones have *limited* computing power, such as low-end or mid-range drones that may benefit from edge computing. Our scenario is similar to the one described in [8], where the drone captures video footage and uses a lightweight onboard *image classification* model to determine frames of interest (e.g., determining if a video frame contains a human subject). After identifying relevant video frames, intensive computation—such as *object detection,* which generates a bounding box around each human subject in the video (or facial recognition, if necessary) –is offloaded to a cloudlet at the edge. The key idea is that end-to-end transmission latency is shorter than performing object detection directly on the drone, which necessitates using a cloudlet in our system. Our proposed scheme is shown in Figure 1.

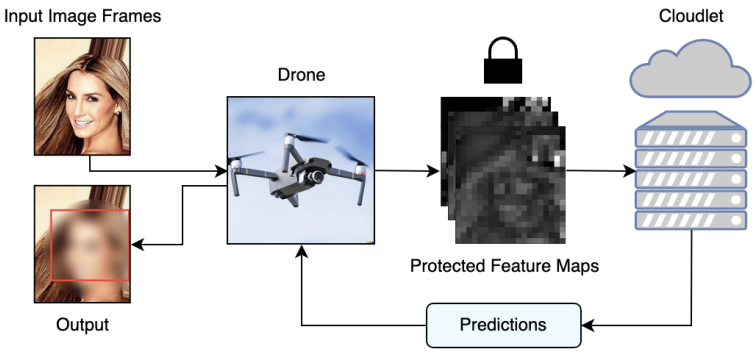


Figure 1. Simplified diagram of core system logistics, showing how the drone receives an input image frame through its camera. The onboard model extracts the feature maps and sends them in a non-invertible form to the cloudlet. The cloudlet sends the predictions (i.e., the location of the bounding box for the human object) to the drone without needing to access the original video frames.

We leverage the commonalities between DNN-based image classification and object detection models to develop an efficient privacy-preserving distributed deep learning model. Image classification models can be used for inference onboard the drone without incurring a significant latency overhead. For example, MobileNetV3 [9] and SqueezeNet [10] are small

models that have fast CPU inference times and are suitable for edge devices. However, downstream tasks like object detection or object tracking are inherently compute-intensive because they require calculating the bounding boxes of objects in addition to classifying the objects in the scene. Rather than choosing a fast but less accurate model solely on the drone, we can use a relatively powerful model in a distributed setup. DNN-based object detection models consist of a backbone network and object detection modules that leverage the features extracted by the backbone network. The backbone network is based on an image classification network (e.g., MobileNet) and has fewer layers, as it is designed to extract features rather than assign class labels. The backbone network functions like an image classification network without the last few layers.

In our system, when the onboard image classification model detects relevant video frames, the drone sends the features extracted from the video frames to the cloudlet for processing. The cloudlet contains additional modules to compute the bounding box and classification predictions, and these modules use the extracted features, rather than raw video frames, as inputs. Thus, any potentially sensitive information in the video frames is not directly exposed to the cloudlet.

The cloudlet is treated as a semi-trusted entity and as mentioned in [2], the cloudlet is an "honest-but-curious" adversary that may try to learn additional information from the images it receives from the drone. The drone is treated as a trusted party within the security model. That is, we do not assume the drone operator is a professional hacker who can change or control the hardware and firmware of the drone in use. Our proposed approach prevents sensitive information from being leaked to the cloudlet while respecting the design goals discussed in the following subsection.

3.2. Design Goals

3.2.1. Privacy

When considering aerial drones in urban environments, personally identifiable information (PII) includes the faces captured in the live video feed and the relevant image features extracted by a model during drone deployment. Sensitive information such as faces can be used by adversaries to derive a database of where people reside, exploit the information for systems that rely on biometric authentication, etc. Protecting image/video features and any other intermediate representations is just as important as protecting the original faces. Extracted features contain semantic information related to the original image frames. These features can be used to reconstruct faces and recover the identities of people in the video feed.

As such, drones tasked with traveling in urban environments will often encounter bystanders, and detecting them may either be the primary goal or a necessary step as part of a downstream computer vision task. The system must guarantee that the frames containing faces and relevant image features remain protected from the cloudlet. We understand that, depending on the extra knowledge of an attacker on a person caught in the camera, information other than faces, e.g., clothes, may also be used to identify the person. However, at this stage, our research only focuses on protecting face-based PII.

3.2.2. (Near) Real-Time Streaming Support

In the previously mentioned tasks, real-time streaming support is required to effectively derive insights from data captured by aerial drones at scale. The actual latency requirements may vary depending on the specific use case. For example, emergency response drones should be able to deliver low-latency video streams with a strict upper bound, whereas delivery drones may not have such stringent requirements but should still provide video streams with an acceptable delay.

3.2.3. Bandwidth Efficiency

Systems for video analysis using drones need to be designed with bandwidth efficiency as a key consideration in order to be scalable. Each drone streams high-definition data,

which when not managed properly, can quickly saturate network bandwidth. The goal is to design a system that can seamlessly accommodate as many drones as possible with minimal performance degradation.

Not all data captured by drones are useful. For video analytics, drones need to be equipped with onboard models or heuristics that can determine if some footage is important enough for transmission. The bandwidth usage can be further optimized using data compression techniques, which may be dependent on the nature of the specific use case.

3.3. Challenges

Existing work has shown that each of the aforementioned design goals is feasible, but designing a system that finds a balance and achieves the goals in tandem is challenging. Some key challenges to consider are as follows:

- Classical computer vision algorithms are fast and suitable for real-time streaming but have relatively low accuracy compared to DNN-based methods. While there are DNN models built for devices with low computing power, there is much to be explored in leveraging edge computing for tasks that require intensive computation.
- Edge computing has alleviated the bottleneck in data processing for deep learning, but it has also introduced a new set of privacy challenges. The exposure of sensitive data to the cloudlet needs to be minimized, though this often comes at the cost of increased latency due to preprocessing or reduced utility.
- Popular object-tracking methods in videos rely on known kernels or approximate the location of objects of interest to optimize computation and execution time. However, this may result in the leakage of PII, which may nullify any desired benefits.

4. Methods

In this section, we present an overview of our system, discuss the design choices for the neural network, and the modules used to prevent the leakage of sensitive information from the drone's transmitted data.

4.1. Overview

4.1.1. Neural Network Architecture

In order to support real-time streaming requirements, we limit our focus to one-stage object detection frameworks such as SSD [11] and YOLO [12]. Architectures such as Faster R-CNN [13] can achieve high accuracy in general-purpose object detection tasks, but the region-based approach in Faster R-CNN and other two-stage object detection frameworks is unsuitable for mobile devices with limited computational power.

One-stage object detection architectures use a proposal-free approach to make predictions with impressive accuracy and speed, which is suitable for mobile and edge devices. They typically consist of a convolutional neural network (CNN) and an object detection meta-architecture such as SSD. The CNN is based on a standard architecture used for image classification and is known as the backbone architecture. The SSD method aggregates information from feature maps extracted by the CNN at different scales to generate predictions about objects. We choose to use MobileNetV3 [9] as the backbone architecture because of its impressive speed on devices with limited computing power and couple it with the SSD meta-architecture for real-time object detection.

In addition to supporting real-time streaming requirements, we enforce a privacy-preserving approach by implementing the SSD-MobileNetV3 model in the form of a *split-model architecture.* As the name suggests, the object detection model is split across the drone and the cloudlet. The motivation for pursuing this approach stems from the fact that PII can be leaked from the raw video frames transmitted to the cloudlet in a traditional compute-offloading scenario. The SSD head in the model only requires the feature maps extracted at different scales, which are intermediate representations (IRs) of the neural network. While feature maps can contain edge information, overall shape, and some invariants, they do not

contain PII explicitly. In the split-model architecture, the drone receives video footage as input and generates the feature maps for relevant video frames. The cloudlet receives the feature maps as input and generates predictions, which are sent back to the drone without any PII being explicitly leaked.

4.1.2. Feature Maps

Feature maps (also known as *activation maps*) are intermediate results obtained as an input image is processed through the several layers of a CNN. Each convolutional layer is composed of a set of filters with learned weights that extract a feature map per filter. Early stages of the network extract local information such as edges and corners, which may be present in the image. The visualizations of these early-stage feature maps tend to resemble the original image to some extent. However, feature maps extracted at deeper layers of the network lack resemblance with the original image as they contain relatively abstract information. Furthermore, spatial information is lost as the image is processed through the network. We try to leverage this property of feature maps in CNNs to maintain privacy in a compute-offloading scenario.

Distributed deep learning systems leverage compute offloading due to the ubiquitous connectivity supported by modern wireless systems. From the perspective of the drone, the cloudlet has much richer computational resources. In the split-model architecture, the feature maps or IR are sent to the cloudlet for further processing. While the feature maps do not contain PII, it is necessary to consider the possibility of recovering PII from the feature maps through an adversarial attack. Recovering (or partially recovering) PII from the IR is an inverse problem and an image reconstruction attack is possible using, for example, an autoencoder or GAN, despite the challenges that modern DNNs pose.

4.1.3. Additional Modules

To engineer a robust solution resistant to reconstruction attacks, we make use of a *selector module* and a *noise module,* as shown in Figure 2. The noise module incorporates additive noise sampled from a Gaussian distribution into the feature maps while the selector module annuls a subset of feature maps based on predefined policies. Recovering sensitive information from only a subset of the feature maps is significantly harder than reconstructing the images using all the feature maps.

In the noise module, there are two hyperparameters to control the noise injected: (i) Variance of the Gaussian noise (σ^2), and (ii) mean of the Gaussian noise (μ). For a set of feature maps $\mathbf{x} \in \mathbb{R}^{B\times C\times H\times W}$, the set of perturbed feature maps $\mathbf{x}'$ is given by the following:

$$\mathbf{x}' = \text{clip}(\mathbf{x} + \delta, 0, 1),\ \delta \sim \mathcal{N}(\mu, \sigma^2)$$

Here, B is the batch size, C is the number of feature maps, H is the height of the feature maps, W is the width of the feature maps, and $\delta \in \mathbb{R}^{B\times C\times H\times W}$ is the additive noise. Since $\mathbf{x}'$ substitutes $\mathbf{x}$ as an intermediate representation, we clip the values to [0,1] to prevent gradient vanishing or explosion when the model is trained.

In the selector module, the fraction of feature maps to be annulled is given by the hyperparameter λ. A naive annulment can be implemented by simply sending a subset of the feature maps to the cloudlet, however, for each distinct value of λ, the model architecture on the cloudlet will need to be changed. Consider a subset of feature maps $\mathbf{x_s} \subset \mathbf{x}$ such that $\frac{|\mathbf{x_s}|}{|\mathbf{x}|} = \lambda$, and assume that the feature maps $\mathbf{x_s}$ need to be annulled. Then, the three annulment strategies are as follows:

(i) Setting the feature maps to 0 ($k = 0$): We set $\mathbf{x_s} = \mathbf{0}$ i.e., the feature maps to be annulled have all their values set to 0. For example, if one 20×20 feature map is to be annulled, all 400 values of the feature map are set to 0.

(ii) Setting the feature maps to 1 ($k = 1$): We set $\mathbf{x_s} = \mathbf{1}$ i.e., the feature maps to be annulled have all their values set to 1.

(iii) Setting the feature maps to be noise ($k \sim \mathcal{N}(0,1)$): In this case, each value of the feature map is sampled from a unit normal distribution.

By using the aforementioned strategies, the cloudlet model architecture can remain unchanged allowing for greater flexibility.

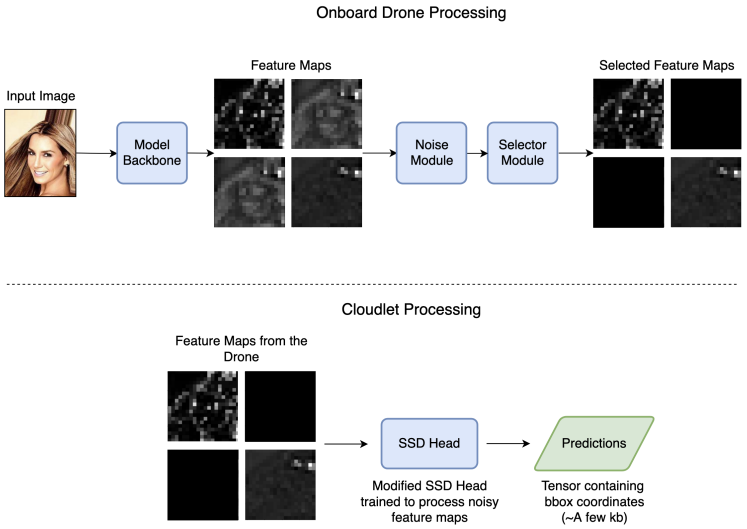


Figure 2. The onboard *Model Backbone* module extracts feature maps. The *Noise Module* injects noise from $\mathcal{N}(\mu, \sigma^2)$, and the *Selector Module* chooses a subset of the feature maps. The drone sends the processed feature maps and additional metadata to the cloudlet for further processing.

Intuitively, one could question the use of noise injection to feature maps instead of the raw image. Adding noise to the original image does not provide sufficient privacy protection. Instead, Gaussian noise can make it easier for adversaries to reconstruct images because it is a powerful data augmentation technique for image reconstruction [14]. The amount of noise that needs to be added to the raw image to preserve privacy would severely degrade model performance. Thus, we add noise to the feature maps to strike a balance between performance and privacy protection.

4.2. Bandwidth-Efficient Streaming

In our system, there are two aspects that need to be considered for bandwidth efficiency: (i) the size of network packets sent from the drone, and (ii) the frequency of transmission. We follow a similar philosophy mentioned in [8] to control the frequency of transmission. The drone sends data to the cloudlet only if the onboard image classification model detects an object of interest (e.g., faces in our application). Furthermore, we can reduce bandwidth usage by sending compressed features instead of raw video frames since the network packet containing feature maps is smaller than the raw video frames. In our approach, we encode the feature maps as PNG images. The packet sent by the drone consists of extracted features and meta-data about the video stream (e.g., frame dimensions).

4.3. Object Tracking and Real-Time Streaming

Object tracking is an extension of object detection in the domain of videos, and as such we leverage our proposed object detection model as part of the tracking algorithm. However, running our object detection model on each video frame is a naive approach that is rather inefficient in terms of bandwidth efficiency and latency. Typically, object-tracking algorithms require an initial prediction (bounding box) for the first frame and the positions of relevant objects throughout the subsequent frames are estimated by exploiting the spatiotemporal locality in videos. New predictions need to be periodically generated in order to minimize the error in the tracking algorithm. The key idea is that the estimation

process of object tracking is significantly faster than repeated object detection inference while being sufficiently accurate.

Thus, to support a low-latency bandwidth-efficient framework suitable for real-time video analytics, an appropriate amount of frame sampling needs to be incorporated. Instead of transmitting the features of every video frame, the onboard model transmits the features of only a fraction (e.g., 10%) of all the video frames. Thus, if the camera on the drone captures video footage at 10–30 frames per second, the drone transmits the features of only 1 to 3 frames in a second. The video frames selected for transmission are uniformly sampled from all the video frames recorded. In our system, we implement a lightweight object-tracking algorithm on the drone and periodically query the cloudlet for object detection predictions. The tracking algorithm is local to the drone for two reasons: (i) to maintain bandwidth efficiency by eliminating constant transmission of bounding box information to the drone, and (ii) to preserve privacy because the cloudlet cannot access the raw video frames necessary for object tracking.

5. Implementation

In our implementation and experiments, we use identical hardware resources and a WiFi connection to resemble real scenarios as best as possible.

Specifically, we use Docker (v.20.10.23) [15] to emulate a 6-core (2 GHz) CPU with 8 GB of RAM and no GPU for our drone platform since we assume the drone is equipped with limited computing power. In comparison to desktop and server-rack GPUs, the total available RAM and memory bandwidth are significantly lesser on the drone. The cloudlet is emulated using a single NVIDIA T4 GPU, which is an entry-level GPU supported by most cloud providers including AWS and GCP. The Tesla T4 GPU is optimized for AI workloads and is sufficiently powerful to demonstrate the effectiveness of our system. To emulate the actual wifi communication between a drone and a cloudlet, we also use wifi for the transmission of feature maps and predictions between them, using PyZmq (v.26.2) to implement the messaging protocol. When we deploy the system, the onboard and cloudlet models are run in a client-server manner. As a result, the inference time of a split model includes the time taken to extract features, encode features and metadata into a packet, transmission time (to and from the drone), and the time taken to compute bounding boxes using the cloudlet model. Docker allows near-perfect emulation of lower-end drone capabilities since CPU cores, RAM, ports, etc. can be precisely allocated.

6. Results

In this section, we present results that validate the feasibility of the split-model architecture for object detection and object tracking while preserving visual privacy. The evaluation setup, evaluation metrics, results, and discussion are presented in the following subsections.

6.1. Evaluation Setup

Our proposed object detection scheme is benchmarked on the PennFudan [16] dataset, which is a pedestrian dataset. These datasets are particularly relevant to detecting humans in images as opposed to traditional object detection benchmarks such as MS-COCO or ImageNet.

To evaluate how well our system preserves privacy, we present image reconstruction results on the CelebA [17] and PennFudan datasets. We use the MOT16 [18] dataset to benchmark our object-tracking model. The goal is to perturb the feature maps with minimal degradation in system performance. To this end, we study the quality of the perturbed feature maps based on different hyperparameter configurations and analyze the quantitative and qualitative results.

6.2. Evaluation Metrics

In this subsection, we will discuss the relevant metrics used to evaluate our system for different tasks. Additionally, we discuss how these metrics should be interpreted in the context of privacy and the intuition behind the expected trends. In the case of object detection, we use precision and recall at specific intersection over union (IoU) thresholds. We can summarize these metrics and their relevance to privacy as follows:

- Intersection over union (IoU): In object detection, each of the input images has target or ground truth bounding boxes, which are the true locations of the objects in the image. Ideally, the predicted bounding box should have maximum overlap with the ground truth and be of similar proportions. A high IoU score for a predicted bounding box suggests high accuracy.

$$IoU = \frac{\text{Common area}}{\text{Total area spanned by both boxes}}$$

- Precision: It measures the reliability of the positive predictions made by the model. In object detection, a prediction is "positive" if the IoU score for the prediction is greater than a predefined threshold. For example, AP@0.5 is interpreted as the average precision at the IoU = 0.5 threshold i.e., if the IoU score for a target-prediction bounding box pair exceeds 0.5, it is considered a positive prediction, and negative otherwise.

$$Precision = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

  If a bounding box prediction is made but the IoU score is below the chosen threshold, it is considered a false positive. Even though it is undesirable, a false positive will only lead to the protection of a nonsensitive area; that is, privacy is not affected. Thus, while we want precision to be as high as possible, we note that reduced precision is not much of a privacy concern.
- Recall: It measures how well the system can identify all relevant objects in the video frame. This is particularly important in the context of privacy because false negatives are objects that the model was not able to detect. Undetected objects will not be blurred and can potentially leak sensitive information.

$$Recall = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

6.3. Evaluation of Privacy Protection

In this section, we evaluate the privacy protection offered in our proposed framework by studying different hyperparameter configurations. Our goal is to find a configuration that offers the best performance while preventing the exposure of sensitive information to the cloudlet. In the following experiments, we evaluate our model based on how well the cloudlet can reconstruct the original video frames using the feature maps sent from the drone.

We model the cloudlet's attempt to learn sensitive information as an *image reconstruction attack*, with the threat model illustrated in Figure 3. The attack is implemented using an Autoencoder architecture. The attack results are shown in Figure 4 and they are based on the methodology in Figure 5. The autoencoder model comprises an encoder that learns a latent representation of the input (the raw video frames/images) and a decoder that can reconstruct the original input. During training, we choose the drone's backbone feature extraction model as the encoder, which generates the feature maps sent by the drone as the latent representation. The decoder is an adversarial model that the cloudlet uses to learn sensitive information. Here, we freeze the encoder because the adversary (cloudlet) does not have access to the parameters of the onboard model.

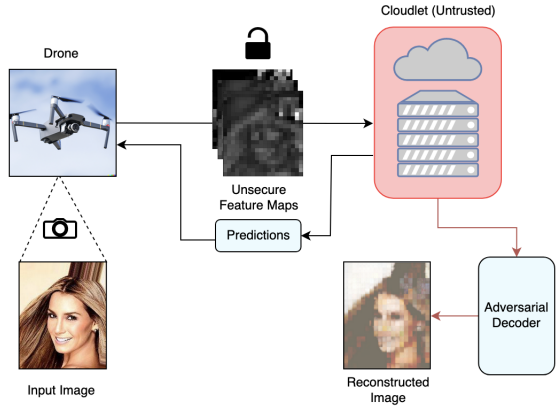


Figure 3. Simplified attack diagram exploiting unprotected features.

Figure 4. Comparison without our proposed scheme. The first column shows original images from the PennFudan dataset. The second column shows images reconstructed using unprotected feature maps (without our scheme), and the third column shows images reconstructed with an annulment factor of $\lambda = 0.3$, noise variance of $\sigma^2 = 0.4$, and mean of $\mu = 0.1$. The annulment value used is based on a Gaussian distribution (i.e., $k \sim \mathcal{N}(0, 1)$).

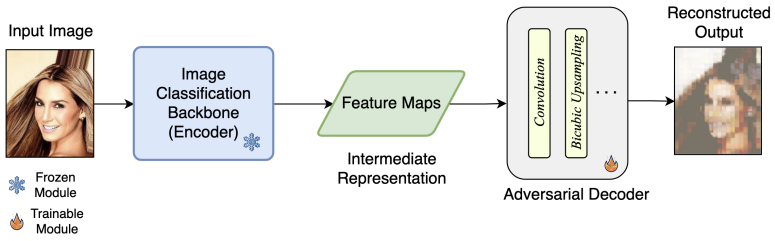


Figure 5. Training methodology for the adversarial decoder.

In this setting, we assume that the cloudlet has access to image–feature map pairs based on the onboard drone model similar to a black-box scenario.

For the CelebA dataset, we train the adversarial decoder from scratch with a learning rate of 1×10^{-4} and a batch size of 64 with a weight decay of 1×10^{-5} for 7 epochs. The results can be reproduced with sufficient accuracy even when using approximately 5000 training images (including augmentation). For the PennFudan dataset, we do not train the model from scratch since the number of samples is too small ($\sim$500). Instead, we fine-tune the model on the dataset using the ImageNet pre-trained weights for 20 epochs. We use Gaussian noise ($\sigma^2 = 0.15$, $\mu = 0.05$), horizontal flips, and vertical flips for data augmentation.

We conduct both *quantitative* and *qualitative* evaluations of the effectiveness of the reconstruction attack. The quantitative measures are useful for observing trends for hyper-parameter configurations, but we also present a qualitative analysis of our system because the existing metrics are not designed with privacy in mind.

The quantitative results are presented for the CelebA dataset and the model configurations are evaluated based on two classic metrics that are known for measuring image similarity: the Structural Similarity Index (SSIM) [19] and Multiscale Structural Similarity Index (MS-SSIM) [20]. In Figures 6 and 7, the impacts of the annulment factor (λ) and noise variance (σ^2) on SSIM and MS-SSIM are shown.

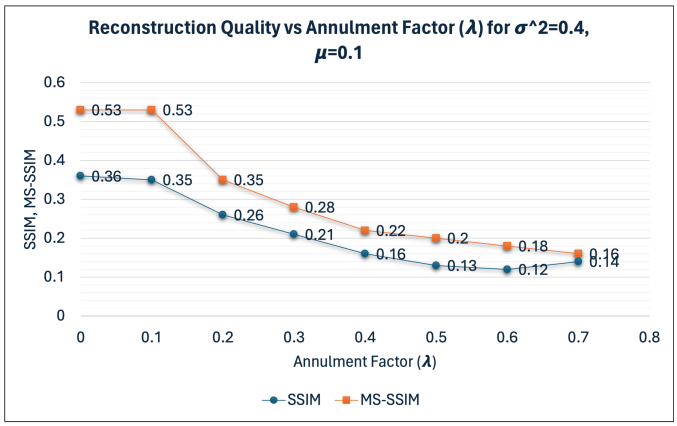


Figure 6. Reconstruction quality (SSIM, MS-SSIM) vs. annulment factor (λ) for a victim MobileNetV3 backbone model on the CelebA dataset.

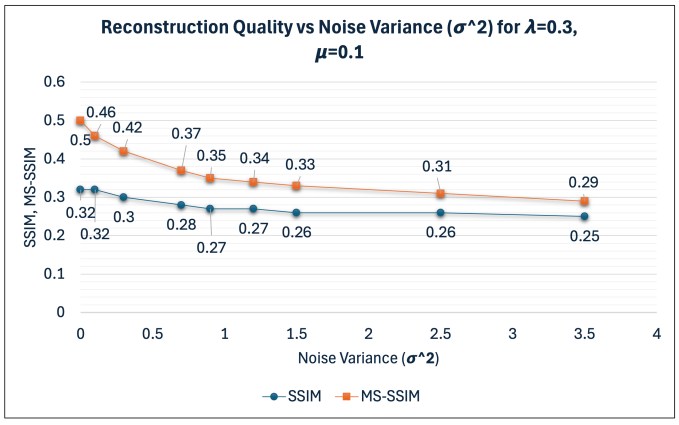


Figure 7. Reconstruction quality (SSIM, MS-SSIM) vs. noise variance (σ^2) for a victim MobileNetV3 backbone model on the CelebA dataset.

In Figure 6, we see that the reconstruction quality is inversely proportional to the annulment factor (λ). We choose $\lambda = 0.3$ as the annulment factor because it is the smallest value that offers sufficient protection against reconstruction from the cloudlet. Ideally, we want the SSIM and MS-SSIM scores to be as low as possible to preserve privacy.

In Figure 7, the reconstruction quality decreases as the noise variance (σ^2) increases, and the reconstruction quality plateaus for values $\sigma^2 \geq 1.3$. We choose $\sigma^2 = 0.4$ because it is the smallest perturbation that offers the best privacy before the values plateau. We observe that the annulment factor has a greater influence on the quantitative analysis of reconstruction quality when compared to the noise variance.

The previous quantitative measures are useful for observing trends for hyperparameter configurations; next, we also present a qualitative analysis of our system because the existing metrics are not designed with privacy in mind. Specifically, we consider some examples of reconstructed images from the PennFudan and CelebA dataset using hyperparameter configurations based on the trends presented earlier. We verify that the observed trends are reflected qualitatively and study observations that are not represented using standard metrics such as SSIM and MS-SSIM.

In Figure 8, we consider the worst-case scenario in which the reconstruction from unprotected feature maps results in images that bear a significant resemblance to the original images. In the third column, we see that our proposed scheme can minimize the exposure of any information in the images to the cloudlet.

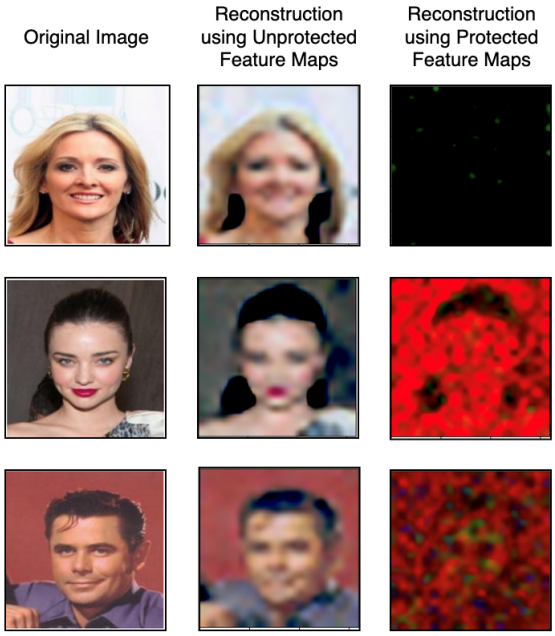


Figure 8. Comparison of reconstruction quality with and without our proposed scheme. The first column shows original images taken from the CelebA dataset, the second column shows the images reconstructed using the unprotected feature maps without using our scheme, and the third column shows the images reconstructed using an annulment factor of $\lambda = 0.3$, noise variance of $\sigma^2 = 0.4$, and mean of $\mu = 0.1$. The annulment value used is based on a Gaussian distribution (i.e., $k \sim \mathcal{N}(0, 1)$).

Now, we discuss how the method of feature map annulment using the annulment value (k) affects the reconstruction quality. As seen in Figure 9, for the same variance threshold (σ^2) and annulment factor (λ), feature map annulment using a Gaussian distribution ($k \sim \mathcal{N}(0, 1)$) is the most effective at preserving privacy. As far as smaller details are concerned, feature annulment with $k = 1$ offers the best performance and privacy.

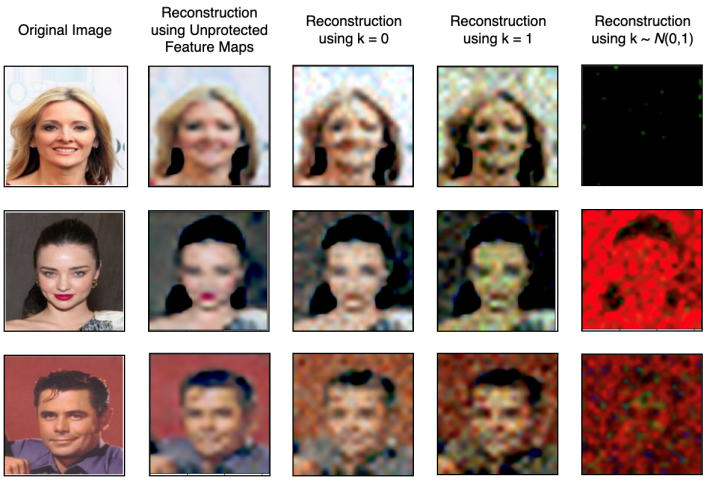


Figure 9. Comparison of privacy protection offered by different annulment values (k) for the variance threshold ($\sigma^2 = 0.4$), annulment factor ($\lambda = 0.3$), and noise mean ($\mu = 0.1$) for the CelebA dataset.

6.4. Evaluation for Object Detection

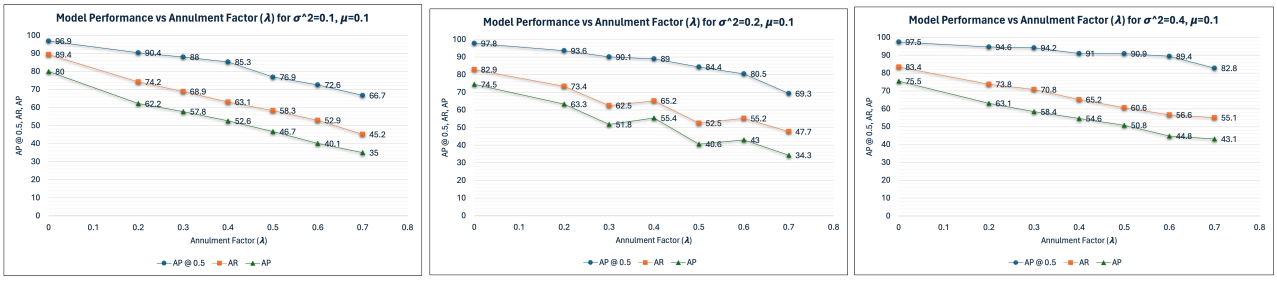
Table 1 shows the object detection results. The noise variance and the mean used in our model are 0.4 and 0.1, respectively. We choose to annul 30% of the feature maps (i.e., $\lambda = 0.3$) in the noise selector module. We observe 3% and 7% decreases in AP @ 0.5 and AR, respectively compared to the baseline. The decrease in recall is because the addition of noise prevents the model from detecting small-scale objects. A typical example is shown in Figure 10(2(b)), where 5 pedestrians on both sides of the camera view missed detection because they are very far away from the camera and, hence, become very small in the view. However, these missed human detections are small enough that they will barely leak sensitive information, even though they are exposed in the frame, as shown in Figure 10(2(c)).

Figure 10. Comparison between SSD-MobileNetV3 (first column) and Secure SSDLite (ours, second column) bounding box predictions for samples from the PennFudan dataset. The yellow bounding boxes are the ground truth and the red boxes are the predicted boxes. The third column blurs the sensitive regions based on the predictions from the Secure SSDLite model with $\alpha = 0.11$. Subfigures 1(a)–3(c) are examples of predicted bounding boxes with varying subjects and sizes.

Table 1. Comparison of Secure SSDLite (our model) against the baseline model for object detection on the PennFudan dataset. Our model uses $\lambda = 0.3$ and $\sigma^2 = 0.4$.

Model	AP@0.5	AR@0.5:0.95
SSD-MobileNetV3	97.4	83.2
Secure SSDLite (Ours)	94.2	76.8

In Figure 11, we study the effect of varying the annulment factor at different variance thresholds for the PennFudan dataset. Upon exploration of the hyperparameter space, we find that our object detection model is far more robust to feature map annulment when $\sigma^2 = 0.4$, as shown in Figure 11b,c. We find the ideal balance between performance and privacy is when $\lambda = 0.3$ and $\sigma^2 = 0.4$. Furthermore, the choice of Gaussian annulment (i.e., $k \sim \mathcal{N}(0,1)$) during feature map selection is confirmed by the results shown in Figure 12.



(**a**) (**b**) (**c**)

Figure 11. Performance of Secure SSDLite (AP@0.5, AP, AR) vs. annulment factor λ for different variance thresholds. Here, the variance threshold refers to the variance, σ^2, of the Gaussian distribution from which the additive noise is sampled. Similarly, the mean, μ, is the mean of the aforementioned Gaussian distribution. (**a**) Performance of Secure SSDLite (AP@0.5, AP, AR) vs. annulment factor λ for variance threshold $\sigma^2 = 0.1$. (**b**) Performance of Secure SSDLite (AP@0.5, AP, AR) vs. annulment factor λ for variance threshold $\sigma^2 = 0.2$. (**c**) Performance of Secure SSDLite (AP@0.5, AP, AR) vs. annulment factor λ for variance threshold $\sigma^2 = 0.4$.

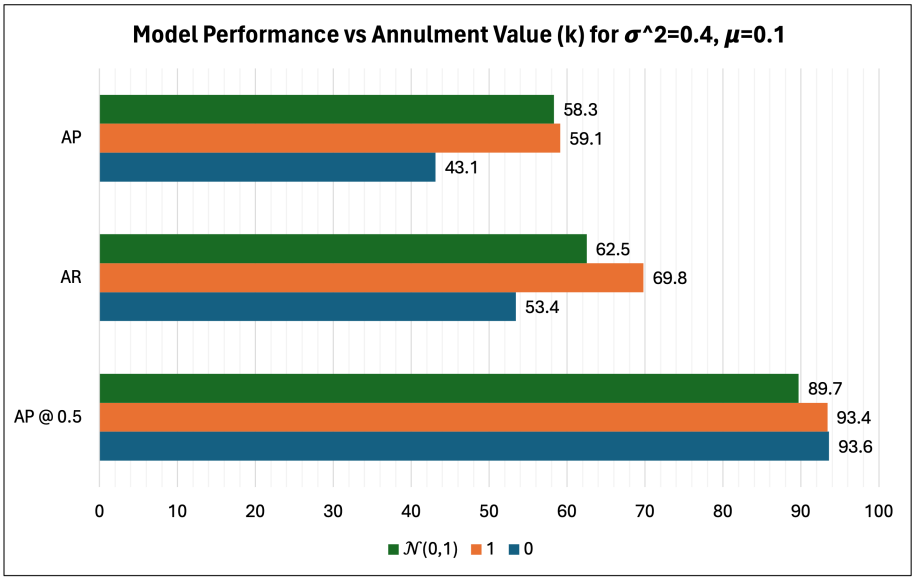


Figure 12. Performance of Secure SSDLite (AP, AR, AP@0.5) vs. annulment value (k) for variance threshold $\sigma^2 = 0.4$.

In Figure 10, we show some example bounding boxes generated using our proposed scheme. We qualitatively evaluate the performance and discuss the privacy implications. In the first row, we consider an image with objects at multiple scales for comparison.

Despite incorporating Gaussian noise and feature map annulment, the pedestrians are localized well in Figure 10(1(b)), but the bounding boxes are larger in 1(b) compared to 1(a). Consequently, the IoU score for 1(b) (bounding boxes generated using Secure SSDLite) is worse than 1(a) (baseline). However, in terms of privacy, there is no issue if the predicted bounding box is slightly larger than the ground truth as long as the person can be blurred without any leakage of sensitive information as seen in Figure 10(1(c)). In the second row, we see that 2(a) contains some very small targets that the model does not detect in 2(b), as mentioned earlier. To prevent leaking sensitive information, we blur a larger area around the bounding box. The additional area to be blurred is determined by a hyperparameter α. Here, $\alpha = 0.11$, which blurs an additional 11% of the area around the bounding box. Thus, our proposed model can effectively localize multiple objects of interest at different scales while preserving privacy.

6.5. Evaluation for Real-Time Applications

A key performance indicator of our system is latency which can support real-time streaming requirements. In Table 2, we present the inference times of onboard feature extraction and cloudlet-side object detection on different device platforms. We see that the feature extraction time is similar to the CPU-only drone and the GPU-powered cloudlet, which aligns with our design choice of running image classification (and feature extraction) on the drone to minimize bandwidth usage. Also, there is a significant overhead (5.9× more than the cloudlet) incurred when running the entire object detection process on the drone. Most importantly, the use of the selector module and noise injection results in only a 2 ms increase in latency.

Additionally, we present the trade-off between bandwidth efficiency and latency due to an additional encoding step. The extracted feature maps sent by the drone are 1.57 MB per video frame (image) without encoding or compression. A large packet size per frame is undesirable for real-time video analytics, and as such we encode the feature maps into images to obtain a 6× improvement as shown in Table 2.

In Table 3, we further examine the overall execution time for all processes on the drone and the cloudlet. In exchange for improved bandwidth efficiency, the drone has a 55 ms overhead to process the feature maps into PNG format, which is a reasonable trade-off considering the significant improvement in object detection latency due to the cloudlet.

Table 2. DNN model profiling and comparison of packet transfer methodologies at frame-level granularity. Here, NVIDIA T4 represents the cloudlet with the GPU.

Device	Feature Extraction Time		Object Detection Time	Packet Size
	MobileNetV3 Baseline (Unprotected Feature Maps)	MobileNetV3 (Protected Feature Maps)	SSDLite Head	
Comparison of Baseline DNN vs. Noise-based Protection without encoding Feature Maps				
Drone	8.20 ms	10.21 ms	101 ms	1.57 MB
NVIDIA T4	6.14 ms	8.38 ms	17 ms	
Comparison of Baseline DNN vs. Noise-based Protection using encoded Feature Maps				
Drone	8.20 ms	65.39 ms	101 ms	265.6 KB
NVIDIA T4	6.14 ms	63.39 ms	17 ms	

Table 3. Breakdown of overall execution time on the drone and cloudlet. Significant penalty incurred when encoding feature maps as PNG images.

Process	Execution Time
Onboard Feature Extraction	8.20 ms
Noise Addition and Feature Map Selection	2.01 ms
PNG Encoding	55 ms
Cloudlet Processing	17 ms

6.6. Evaluation for Object Tracking

In this section, we present the results of object tracking for the MOT16 dataset in Table 4. For each of these experiments, the drone uses the SORT [21] algorithm for tracking objects after obtaining the initial detections from the cloudlet.

Table 4. Performance of object-tracking algorithm using the detection model + SORT. For the Secure SSDLite, with $\lambda = 0.3$ and $\sigma^2 = 0.4$, 1 in every 5 frames is used to generate a new set of predictions from the cloudlet.

Model	Precision	Recall
SSD-MobileNetV3	72.5	57.9
Secure SSDLite (Ours)	70.2	55.8

Here, we see a slight degradation in recall, which is more relevant to privacy. Once again, the overall recall score may seem low due to the presence of several small-scale objects (people very far from the camera) in the scene and during occlusion when the people move out of the camera's field of view. Even though these objects are not detected, they are too small (or lack relevance) to leak any meaningful information to the cloudlet.

Smaller objects in the direct field of view of the camera can still be tracked despite being far away because the estimation of the tracking algorithm can compensate for the inherent limitations of the SSD object detection architecture.

7. Discussion

Our system uses noise and other lightweight methods as deterrents to adversarial image reconstruction. It could be argued that the scope of threats to DNN-based object-tracking and detection models extends beyond reconstruction. For example, adversarial training methods such as FGSM [22] and PGD [23] can be used to craft malicious inputs that jeopardize the model. However, based on our trust model, the adversary has no control over the input video feed to the drone. Furthermore, the adversary cannot access gradients or any intermediate information other than the feature maps transmitted from the drone. Thus, the scope of the attack is limited to image reconstruction attacks. Although autoencoders are commonly used for image reconstruction, adversaries could alternatively employ generative adversarial networks (GANs) [24] to train adversarial decoders. In our future research, we plan to assess how effectively our defensive strategies can mitigate attacks using GANs.

For Secure SSDLite, our proposed object detection model, the key hyperparameters are the annulment factor λ and the variance of noise σ^2. There is typically a trade-off between precision and recall. As shown in Figure 11a–c, increasing λ enhances privacy protection against the cloudlet but compromises recall. If the λ value is too low, the cloudlet can reconstruct sensitive information; conversely, if it is too high, recall suffers, resulting in unblurred individuals in the video frame. Figure 11b demonstrates that with $\sigma^2 = 0.4$, the impact on recall is minimal, allowing for more robust privacy protection without significantly affecting recall. Although a similar trend is observed with precision, our focus remains on preserving recall. Qualitatively, not only are we able to provide complete obfuscation of faces, but we are also able to obfuscate the vast majority of information in the image background and other secondary characteristics that the cloudlet may try to learn.

Our proposed scheme is geared toward an edge-computing scenario in which cloudlets support the drones by processing computationally intensive workloads. In a traditional cloud computing scenario, multiple drones live-streaming video footage can quickly saturate network bandwidth, and thus, relying only on centralized cloud infrastructure is not scalable. As discussed earlier, another issue is poor latency, which is unsuitable for real-time analytics. Thus, edge computing solutions that involve cloudlets alleviate this bottleneck. While our proposed method has been discussed in the context of edge-computing, it can trivially be extended to a cloud-computing scenario where the adversary is the cloud instead of the cloudlet.

In future work, we will aim to explore whether a more fine-grained approach through model pruning can protect against reconstruction while minimizing the impact on model utility. Additionally, we will broaden our definition of PII and consider other potentially sensitive attributes such as addresses, documents, homes, cars, clothing, and peripheral objects in the scene, as well as study how to address the leakages of such features. We also plan to study how our methodology can be adapted to different object-tracking algorithms and non-standard object-detection network architectures.

8. Conclusions

In this paper, we address various challenges, including privacy issues, bandwidth limitations, and real-time streaming complications, which are prevalent in live video analytics on resource-constrained drones. We present a split-model architecture that consists of (i) A Gaussian noise-based injection (noise module) and (ii) a set of feature map selection strategies, to make feature maps sent to the untrusted cloudlet non-invertible. The non-invertible feature maps ensure that the privacy of bystanders encountered during deployment is protected, particularly in reconstructing facial features, while our selection strategy further enhances privacy protection. The combination of our strategies aims to provide strong privacy protection under real-time constraints and the use of feature maps instead of raw images minimizes bandwidth usage. Based on our systematic analysis of design choices and experiments, our noise module provides the best privacy protection when used with the Gaussian annulment feature map selection strategy. Furthermore, encoding feature maps as PNG images significantly improves bandwidth usage with a reasonable latency overhead. In addition to evaluating object detection capabilities, we validate the effectiveness of our method in an object-tracking setting using the canonical MOT16 dataset.

In recent years, drone technology has proven useful in several promising applications, yet its full potential remains untapped. While there are several valid security and privacy concerns regarding the large-scale adoption of drones, our work proposes an idea that is a step in the right direction toward the safe adoption of drone technology.

Author Contributions: Conceptualization, C.W., Y.S., N.K. and S.Z.; Methodology, P.N., C.W. and Y.S.; Software, P.N.; Writing – original draft, P.N. and S.Z.; Writing—review & editing, P.N., C.W., Y.S., N.K. and Y.H.; Supervision, S.Z.; Project administration, S.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author. The datasets used for training our models are all publicly available.

Conflicts of Interest: Author Yuanyi Sun was employed by the company ByteDance Inc. The remaining authors declare that the re-search was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Drones by the Numbers. Available online: https://www.faa.gov/node/54496 (accessed on 4 November 2024).
2. Tian, X.; Zheng, P.; Huang, J. Robust Privacy-Preserving Motion Detection and Object Tracking in Encrypted Streaming Video. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 5381–5396. [CrossRef]
3. Hu, S.; Wang, Q.; Wang, J.; Qin, Z.; Ren, K. Securing SIFT: Privacy-Preserving Outsourcing Computation of Feature Extractions Over Encrypted Image Data. *IEEE Trans. Image Process.* **2016**, *25*, 3411–3425. [CrossRef] [PubMed]
4. Upmanyu, M.; Namboodiri, A.M.; Srinathan, K.; Jawahar, C. Efficient privacy preserving video surveillance. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 1639–1646. [CrossRef]
5. Juvekar, C.; Vaikuntanathan, V.; Chandrakasan, A. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1651–1669.
6. Morel, A.E.; Kavzak Ufuktepe, D.; Ignatowicz, R.; Riddle, A.; Qu, C.; Calyam, P.; Palaniappan, K. Enhancing Network-edge Connectivity and Computation Security in Drone Video Analytics. In Proceedings of the 2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, USA, 13–15 October 2020; pp. 1–12. [CrossRef]
7. Liu, Y.; Ma, Z.; Liu, X.; Ma, S.; Ren, K. Privacy-Preserving Object Detection for Medical Images with Faster R-CNN. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 69–84. [CrossRef]
8. Wang, J.; Feng, Z.; Chen, Z.; George, S.; Bala, M.; Pillai, P.; Yang, S.W.; Satyanarayanan, M. Bandwidth-Efficient Live Video Analytics for Drones via Edge Computing. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, USA, 25–27 October 2018; pp. 159–173. [CrossRef]
9. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. *arXiv* **2019**, arXiv:1905.02244. [CrossRef]
10. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5MB model size. *arXiv* **2016**, arXiv:1602.07360. [CrossRef]
11. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision—ECCV 2016*; Springer International Publishing: New York, NY, USA, 2016; pp. 21–37. [CrossRef]
12. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2016**, arXiv:1506.02640. [CrossRef]
13. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv* **2016**, arXiv:1506.01497. [CrossRef] [PubMed]
14. Xie, Q.; Dai, Z.; Hovy, E.; Luong, T.; Le, Q. Unsupervised Data Augmentation for Consistency Training. In *Proceedings of the Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: New York, NY, USA, 2020; Volume 33, pp. 6256–6268.
15. Merkel, D. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.* **2014**, *2014*, 2.
16. Wang, L.; Shi, J.; Song, G.; Shen, I.f. Object Detection Combining Recognition and Segmentation. In Proceedings of the Computer Vision—ACCV 2007, Tokyo, Japan, 18–22 November 2007; Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 189–199.
17. Liu, Z.; Luo, P.; Wang, X.; Tang, X. Deep Learning Face Attributes in the Wild. In Proceedings of the Proceedings of International Conference on Computer Vision (ICCV), Santiago, Chile, 7 December 2015.
18. Milan, A.; Leal-Taixe, L.; Reid, I.; Roth, S.; Schindler, K. MOT16: A Benchmark for Multi-Object Tracking. *arXiv* **2016**, arXiv:1603.00831. [CrossRef]
19. Wang, Z.; Bovik, A.; Sheikh, H.; Simoncelli, E. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
20. Wang, Z.; Simoncelli, E.; Bovik, A. Multiscale structural similarity for image quality assessment. In Proceedings of the The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, Pacific Grove, CA, USA, 9–12 November 2003; Volume 2, pp. 1398–1402. [CrossRef]
21. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple online and realtime tracking. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468. [CrossRef]
22. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2015**, arXiv:1412.6572. [CrossRef]
23. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv* **2019**, arXiv:1706.06083. [CrossRef]
24. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *arXiv* **2014**, arXiv:1406.2661. [CrossRef]