Low-Latency Private ML Inference for Vision Tasks in a Distributed Environment

Te Yi Kan, Konstantinos Psounis

USC Networked Systems Performance and Design Lab

USC Viterbi School of Engineering
Ming Hsieh Department of Electrical and Computer Engineering

Preliminaries

Overview:
Applications like mixed reality rely heavily on vision models. However, due to the high complexity of these vision tasks and the limited processing power of mobile devices, these models are typically deployed on distributed systems with multiple remote servers to optimize utility and minimize latency. This setup, however, requires users to share personal data with remote servers, posing potential privacy risks. To address this challenge, we propose a system that enables real-time private inference for vision tasks within distributed environments.

Threat Model:
- ***Strong adversary:*** This adversary has access to image datasets with distributions similar to that of the input images. (e.g., input images follow distributions similar to publicly available datasets.)
- ***Weak adversary:*** The weak adversary lacks prior knowledge of the input image distribution, resulting in lower reconstruction quality compared to the strong adversary.

Metric:
- ***Privacy:*** We leverage two privacy metrics, *structural similarity index measure (SSIM)* and *semantic embedding similarity (SIM),* to assess the quality of the reconstruction of the sensitive parts of the image.
- ***Utility:*** We propose to examine the utility degradation introduced by the obfuscation system.
- ***Latency:*** We evaluate the additional latency introduced by the privacy-preserving process**.**

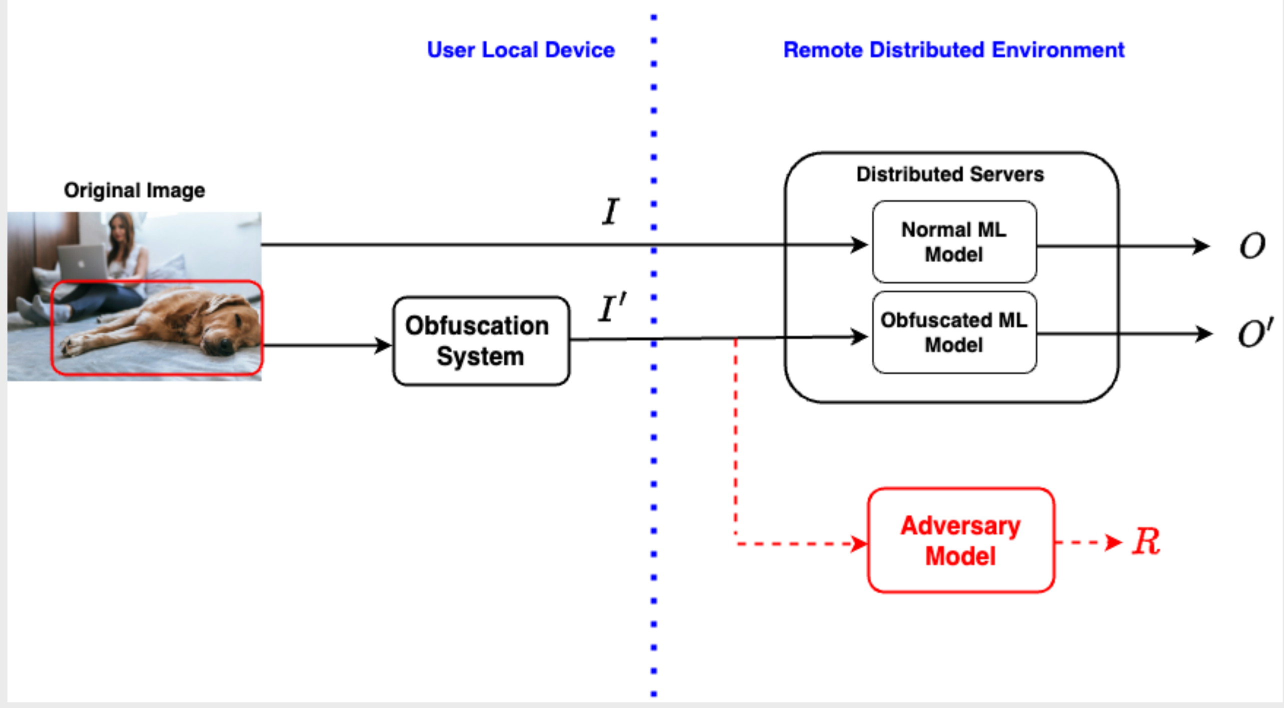

Figure 1: Problem Overview

System Design

Key Insights about how to trade between privacy, utility, and latency:
Not the entire image affects utility, and not the whole image is vulnerable to privacy leaks. The obfuscation system must perform differently on various parts of an image, as shown in Table 1.

Proposed System:
The proposed low-overhead privacy-preserving system consists of a sensitive object detector → scheduler → obfuscator, and they work as follows:

1) Sensitive object detector: A lightweight object detector is incorporated to identify sensitive objects in the input image.
2) Scheduler: The scheduler partitions the input image into multiple chunks and forwards them to various servers to exploit the overall computation capacity in a distributed environment.
3) Obfuscator: The obfuscator first embeds the sensitive images into their latent space embeddings. Afterwards, the mask generation network in the obfuscator takes the embeddings as input and outputs a mask. The obfuscator then applies the mask to the embedding through channel-wise multiplication. This mask is a vector matching the number of channels in the input embedding and will modulate each channel individually.

The proposed system differentiates the images into different parts, as shown in Table 1, and only applies obfuscation on the image's sensitive parts, which affect utility. As a result, the proposed system can achieve a favorable balance between privacy, utility, and latency.

	Sensitive	Non-sensitive
Inside the RoI	privatize image chunks (Balance among privacy, utility, and latency)	Keep original image chunks (Maximize utility minimize latency without hurting privacy)
Outside the RoI	Discard image chunks (Maximize privacy and minimize latency without hurting utility)	Discard image chunk (Minimize latency without hurting utility)

Table 1: Trade between privacy, utility, and latency

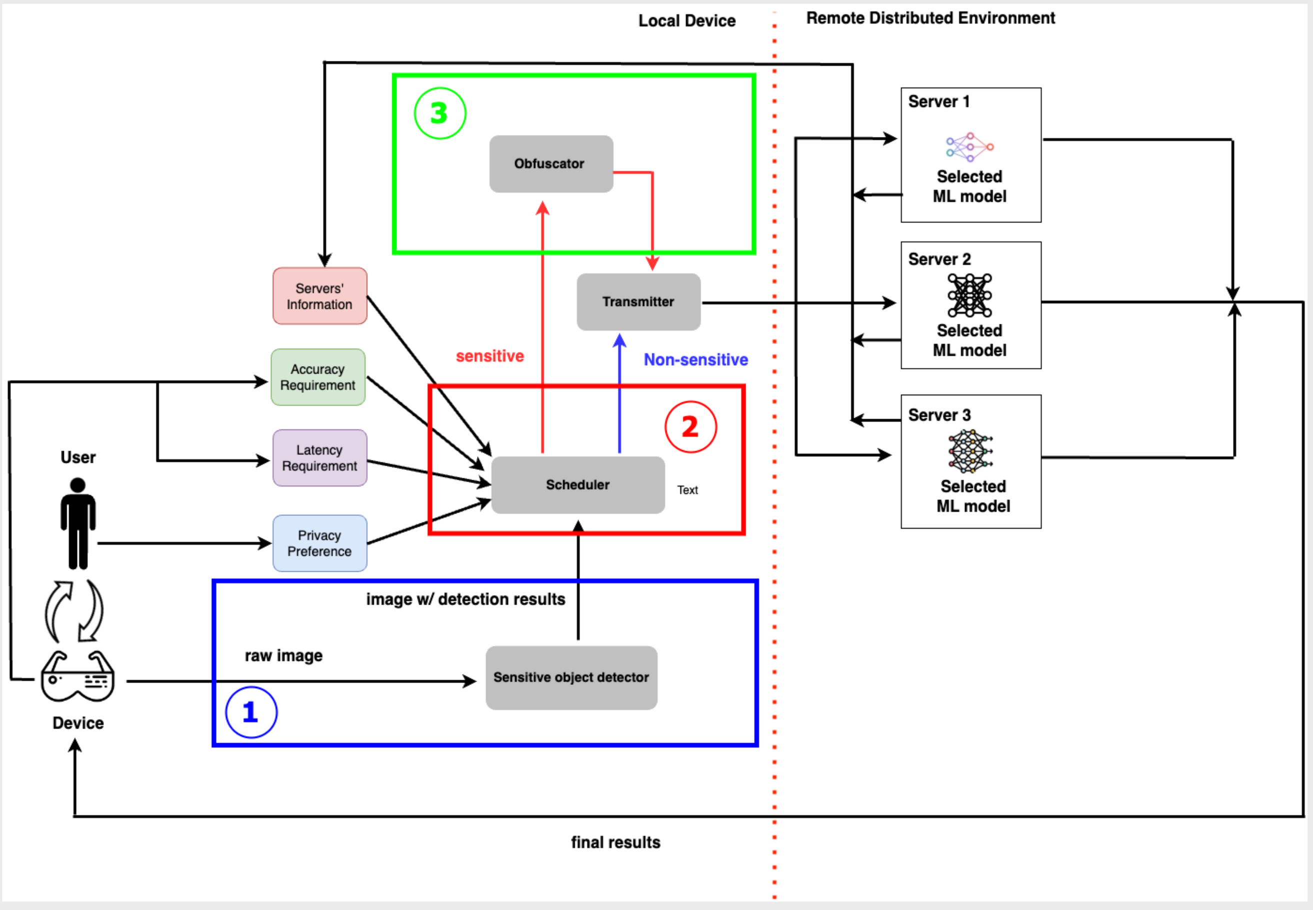

Figure 2. System Model

Evaluation

Table 2: Privacy under a strong adversary with object detection as the target task

Obfuscator	SSIM	SIM	
		ResNet50	VGG16
Proposed	0.116	0.294	0.108
Unaltered Features	0.191	0.303	0.112
Gaussian Noise	0.120	0.295	0.107
Random Mask	0.147	0.300	0.109
PCA embedding	0.124	0.295	0.108

Table 3: Privacy under a weak adversary with object detection as the target task

Obfuscator	SSIM	SIM	
		ResNet50	VGG16
Proposed	0.169	0.348	0.203
Unaltered Features	0.451	0.449	0.313
Gaussian Noise	0.169	0.350	0.201
Random Mask	0.261	0.421	0.242
PCA embedding	0.178	0.371	0.231

Table 4: Utility degradation percentage with object detection as the target task

Obfuscator	Utility (mAP) Degradation
Proposed	11.11%
Unaltered Features	0%
Gaussian Noise	50.40%
Random Mask	36.11%
PCA embedding	55.56%

Table 5: Latency increment with object detection as the target task

Obfuscator	Latency Increment (msec)
Proposed	4.7
Unaltered Features	2.2
Gaussian Noise	$\sim 10^{-3}$
Random Mask	2.2
PCA embedding	$\sim 10^{-2}$