Context-Aware Hybrid Encoding for Privacy-Preserving Computation in IoT Devices

Hossein Khalili *Graduate Student Member, IEEE*, Hao-Jen Chien *Graduate Student Member, IEEE*, Amin Hass *Member, IEEE*, and Nader Sehatbakhsh, *Member, IEEE,*

***Abstract*—Recent years have witnessed a surge in hybrid IoT-cloud applications where an end user distributes the desired computation between the IoT and cloud nodes. While achieving significant speed up, the major caveat of this approach is data privacy. Privacy-preserving methods have received major attention in the past few years, mainly because they can potentially solve this issue. Among several proposals, methods based on dynamic encoding and perturbation offer flexibility and low overhead. However, they often consider a *weak* adversary model or overlook practical limitations such as encoding latency and complexity. This work proposes a new privacy-preserving method to address these issues. The key contributions of this paper are twofold. First, unlike state-of-the-art, it proposes a new approach based on evolutionary algorithms to systematically evaluate the robustness of the encoding algorithm against a large population of potential adversaries. Second, it develops a *dynamic* obfuscation strategy that balances latency requirements in a realistic IoT-cloud hybrid ecosystem and privacy demands. Additionally, our method offers a unique benefit: it can be used alone for privacy protection, or it can be integrated with most existing methods to enhance privacy and reduce latency. The applicability and effectiveness of our proposed methods are thoroughly evaluated using two popular deep neural networks in a real-world IoT-cloud setting. We study the impact of our approach on important metrics, such as accuracy and privacy. Our results show that our proposed method can improve the overall privacy of a given IoT-cloud hybrid ecosystem by more than 10% on average.**

***Index Terms*—Privacy-Preserving, IoT-cloud hybrid systems, machine learning**

I. Introduction

The increase in popularity of IoT devices, the growing availability of low latency network connections, and large investment in cloud servers collectively have increased the popularity of collaborative IoT-cloud computation platforms significantly [1], [2]. In such an ecosystem, the IoT devices offload their data to cloud servers for computation, using a low-latency network (e.g., 5G, WiFi). Such offloading greatly improves the end-to-end latency as the majority (or even all) of the computation is done on a high-end high-performance cloud server rather than the resource-constrained IoT device. This collaboration also improves energy efficiency on the IoT device since it only does the minimal work required to offload the computation instead of locally computing it [1].

Among various popular applications of collaborative IoT-server computation is machine learning, particularly deep learning applications. Recent advancements in deep learning have allowed users to achieve high accuracy in a new class of tasks including vision, natural language processing, and voice recognition. Leveraging a collaborative ecosystem, further allows IoT end-users to leverage these new machine learning capabilities even when a low-end resource-constrained IoT device is being used [3], [4]. The main concern in this ecosystem, however, is security and privacy [5]–[9]. As IoT devices become more common, their reliance on remote servers brings a growing concern for privacy.

To alleviate this quandary, privacy-preserving computation has received lots of attention recently [10]–[22]. These methods allow cloud computing while protecting privacy. Getting the right balance between privacy, inference accuracy, and end-to-end latency, however, remains a key challenge.

An emerging method in this area is *hybrid encoding* methods [7], [21]–[24]. The key idea in a hybrid method is to first *divide* the original model (typically a deep neural network) into two parts, one residing in the IoT device, and the other (rest of the network) staying in the server (see Figure 1a). The network is then *retrained* with the goal of instructing the IoT device to remove sensitive aspects of the end-user data without affecting the other aspects. This could improve the privacy-accuracy tradeoff, as only the private information would be obfuscated, ideally [24].

Motivation. While various hybrid methods have been proposed in prior work [6], [7], [10], [23], [25]–[28], *none has considered the potential challenges that this approach could create on a resource-constrained IoT device*. Particularly, to ensure privacy, the IoT device becomes responsible to detect and remove sensitive information from the raw input data. In other words, hybrid methods shift the burden of encoding to the IoT device. Thus, one of the main challenges in this field is finding privacy solutions that can handle complex networks and tasks without causing a significant increase in overall latency and energy usage.

The second issue motivating the need for our new approach is that existing work often underestimates the adversary's capabilities. This is mainly due to the fact that hybrid methods are primarily inspired, and based upon, machine learning techniques for removing bias and unfairness [26] and hence only consider a *passive* adversary where the attackers only attack at a single point in the network (i.e., the intersection between IoT and server) and use one or very limited number of strategies to infer private information.

The reality, however, is that the attack-defense scene is con-

Hossein Khalili, Hao-Jen Chien, and Nader Sehatbakhsh are with the School of Engineering, Department of Electrical and Computer Engineering at UCLA, Los Angeles, CA, USA 90095, Email: {khalili, chien, nsehat}@ucla.edu.

Amin Hass is with Accenture LLC. Email: aminhass@accenture.com. (*Corresponding Author: Nader Sehatbakhsh.)*

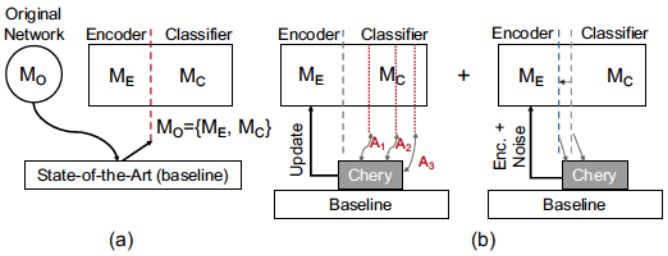


Fig. 1: An overview of two main contributions proposed in our framework, Chery. (a) Existing work uses adversarial training to slice the network into encoder and classifier (M_E and M_C) to protect privacy. (b) Chery improves the state-of-the-art by adding adversary-aware (left) and latency-aware (right) techniques that enhance privacy and latency.

stantly evolving, and hence *the adversary should be modeled as an active and dynamic player who adaptively changes their strategy*. To achieve this, new systematic methods should be developed.

Our Contribution. This work improves the state-of-the-art by proposing a new framework that leverages a *context-aware* design strategy to improve the accuracy-privacy tradeoff while satisfying latency requirements. *Context*, in this paper, refers to practical and realistic considerations such as latency, threat model, and adversary capabilities.

Our proposed solution presents two key contributions. First, to improve privacy, our solution introduces and leverages a new criterion called *adversary-aware*. The main insight is that existing hybrid methods do not properly and extensively model an adversary in their setting, therefore, as will be shown in detail, the achieved privacy decreases. To fix that, we propose multiple new strategies including systematically diversifying the adversary and their capabilities using a metaheuristic evolutionary algorithm. This is in contrast with the existing methods where only a *simple* weak adversary was considered.

Second, to improve latency without sacrificing privacy and/or accuracy, a new *latency-aware* training strategy that *jointly* optimizes latency and privacy is proposed. The key insight is that our algorithm leverages both encoding and perturbation to balance latency and privacy. This is achieved by developing a smart and adaptive perturbation mechanism.

A high-level overview of the key contributions of our work is shown in Figure 1 where (a) shows an existing hybrid method where an adversarial algorithm is used to *slice* the network. On the other hand, Chery adds two new features namely adversary-aware and latency-aware (Figure 1b) which will be described, in detail, in Sections III-D and III-E, respectively. Further, our method can be implemented on top of existing mechanisms, extending their capabilities and improving privacy, latency, and accuracy.

We evaluate our method using two large datasets using two popular convolutional neural networks. We report our results using three important metrics: *latency*, *privacy*, and *accuracy*; and show the impact of each feature introduced in Chery on these metrics. Our results show that privacy can be improved by about 10% on average compared to the baseline. We also compare Chery with various state-of-the-art methods and highlight the advantages of our approach.

In short, the contributions of our paper are as follows:

- We design a new heuristic evolutionary-based algorithm that improves privacy by finding a more powerful, realistic, and diverse adversary.
- We further improve the solution and balance latency and privacy by developing a new dynamic obfuscation algorithm that dynamically employs both encoding and perturbation.
- We implement our framework in a real hybrid IoT-server system and evaluate privacy, accuracy, and end-to-end latency using multiple deep-learning networks and datasets.

The remainder of this paper is structured as follows. In Section II, we discuss the related work. Section III describes the design of Chery in detail. The implementation and results are provided in Sections IV and V, respectively. We briefly discuss other important considerations in our framework in Section VI. The paper is concluded in Section VII.

II. Related Work

For an IoT-server ecosystem, effective privacy-preserving methods allow the end-users to receive the public cloud service while protecting data privacy. To achieve this, privacy-preserving methods provide an effective capability by *obfuscating* their personal information before sending them to the remote server. Such obfuscation comes in various forms including using encryption [19], [29], [30], embedding [23], [25], perturbation [16], [31], and even hardware-support [17], [32]. Instead of obfuscation, another possible solution is based on privacy-preserving truth discovery [33], [34]. The key challenge in all these methods is providing the best tradeoff between privacy and cloud service accuracy while satisfying latency requirements in a collaborative IoT-server ecosystem.

A popular method for obfuscation is homomorphic encryption [19], [29], [35]–[39] which leverages encryption and its unique ability to perform computation over encryption to ensure privacy. The great advantage of homomorphic encryption is its provable privacy guarantees. Similarly, are methods based on other cryptographic primitives, such as garbled circuits and secret sharing [20], [40], [41]. Compared to each other, each method offers various capabilities and is suitable to a particular set of problems (e.g., linear vs. non-linear layers for machine learning computation) [19]. The caveat in these solutions is the notable increase in end-to-end latency. This is caused by the significant computational overhead dictated by using these cryptographic-based methods [39].

A closely related solution is based purely on hardware and particularly trusted execution environments (TEEs) [17], [18], [32]. Methods based on this mechanism rely on the confidentiality and integrity guarantees provided by the hardware enclaves — a special hardware primitive existing in modern systems. Similar to methods based on homomorphic encryption and others, methods based on TEEs also suffer from latency overhead as they are not fundamentally suitable for large-scale and memory-intensive workloads such as deep learning applications.

Also applicable to privacy-preserving computation are methods based on the differential privacy (DP) concept. DP is

a probabilistic privacy mechanism that provides a formal, information-theoretic security guarantee. DP methods have become the de facto standard for protecting private data in databases and machine learning training [15], [42]–[44]. The goal is typically to limit what can be inferred from the model about individual training records. Possible solutions include adding perturbation to gradients, model parameters, and/or the objective function. While existing work shows that DP is quite useful during the training phase, its application to inference is an open question.

Due to the latency issue, there is a need for methods that can provide acceptable privacy without compromising the system's latency, energy efficiency, and other critical aspects. To answer this need, *hybrid* methods have received lots of attention lately [6], [7], [10], [23], [25]–[28]. The key benefit of these approaches is that they largely minimize the computation overhead needed for ensuring privacy. Instead, the network is distributed between the IoT device (client) and the server while the network is also retrained using special algorithms that are designed to ensure privacy.

There are two major shortcomings in the state-of-the-art hybrid methods. Firstly, they are not "resource-aware" as they do not take into account ***practical issues*** such as latency and energy efficiency that arise when an encoding task is delegated to the IoT device. Second, unlike other established security problems existing methods only consider a ***weak*** and stationary adversary, and hence are not "adversary-aware".

III. System Design

A. Overview

We design and implement Context-Aware Hybrid Encoding Privacy-Preserving (CHERY), to provide privacy-preserving capabilities for deep learning applications in IoT-server ecosystems. The main objective of our design is to develop a *context-aware* framework. Particularly, we aim to develop a method that achieves excellent privacy-accuracy tradeoff while being *(a)* adversary-aware and *(b)* latency-aware.

At high-level, our system consists of three major components: *(i)* an *encoder* that resides in the IoT device and is responsible for obfuscation and providing privacy by mapping the input (raw) data to a private *intermediate* representation, *(ii)* a *classifier* that is uploaded into the cloud and is responsible for the main classification task (e.g., image classification), and *(iii)* a *training* strategy/algorithm that is responsible for *(a)* creating and/or distributing the two aforementioned networks and *(b)* designing a mechanism for training each network and for evaluating three important metrics, privacy, accuracy, and latency. Fig. 1a shows this structure at the high level where an original network is divided into an encoder and classifier using an algorithm (called "baseline").

B. Technical Challenges

There are two main technical challenges in designing our method. We describe them in the following:

- **Jointly optimizing accuracy-privacy-latency:** A key challenge for hybrid-based privacy-preserving mechanisms is finding the right balance between competing metrics, namely inference accuracy, data privacy, and overall latency. For more privacy, a more complex encoder is desired, which in turn impacts the latency. Similarly, a stronger encoding could help privacy but it would negatively impact the accuracy (i.e., sending less information changes both privacy and accuracy). We explain how Chery solves this problem by proposing a *context-aware* algorithm.
- **Modeling the adversary:** The other main challenge for hybrid methods is the lack of proper modeling of the adversary. Incorrect assumptions about the adversary could result in a false sense of privacy, hence care should be taken during the design. We explain our *adversary-aware* strategy to address this issue.

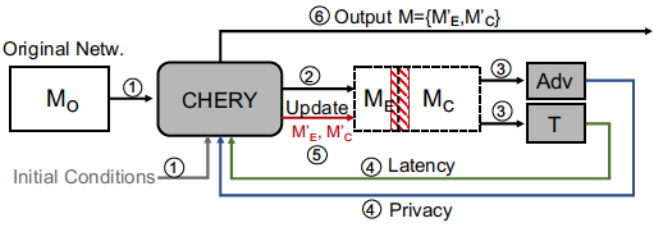


Fig. 2: Steps taken in our design for developing Chery and outputting a new context-aware privacy-preserving model.

C. System Architecture

The systems analyzed in this work are deep neural networks that are used for classification. In Section IV, we will describe the details of networks used for evaluations. In any of those networks, there are two major components in our framework: an *encoder* and a *classifier*.

The overview of design steps for Chery is shown in Fig. 2. Briefly, to design Chery, we take an existing adversarial training strategy (we call it "baseline"), and build our method on top of that. Knowing the baseline training strategy, we then *slice* the original network into two parts: encoder and classifier. In the first step, we slice this network at a *randomly* chosen layer. Due to practical reasons, we limit the random number, r, to be smaller than $L/2$ where L is the total number of layers in the given network. In other words, we limit the slicing such that the encoder is always less than half of the network. This is due to latency constraints. In Section III-E, we describe this constraint in more detail.

There are two fundamental options for the encoder and classifier networks. The first option is to assign the original DNN to the classifier (e.g., Resnet, VGG, etc.) and then design an encoder by introducing new additional layers that are added to the beginning of this classifier. The second option is to *divide* the original network and use the initial layers as the encoder while the rest completes the classification task.

Our analysis shows that the second option has several advantages. First, existing high-performance classifiers are highly optimized and fine-tuned thus adding additional layers, especially at the beginning, completely disrupts the feature extraction functionality of these networks and hence significantly reduces the accuracy. Dividing the network, on the other hand,

does not impact the original functionality of the network and therefore has minimal impact on the accuracy.

Second, adding additional layers at the beginning not only hurts the accuracy but also introduces *additional* latency as more computation is added to the overall system. Adding new layers for the encoder, however, decouples the design of the encoder from the classifier and hence provides flexibility.

Using this strategy, the fundamental question becomes *what is the optimal slicing?* To find the best slicing, we first need to define the *baseline* training strategy (we call it baseline since it does not support context-aware features which will be described later).

Chery leverages *adversarial training* as the baseline strategy as it has become an effective method for privacy-preserving recently [6], [25], [26]. In short, adversarial training in the context of privacy-preserving is an optimization problem that optimizes the tradeoff between two competing objectives: accuracy and privacy.

There has been an extensive body of work on developing an effective adversarial algorithm for privacy-preserving. The main difference between them is how the problem is formulated and how privacy is measured. The two main themes are either based on *(a)* leveraging a *proxy* network to measure privacy (i.e., it is used as a discriminator to evaluate the success of encoding) or *(b)* measuring the mutual information between the encoded vs. the private information. In both cases, the underlying goal is the same. Reward the model if it can successfully remove sensitive information and penalize it if this removal hurts accuracy. To balance the two competing objectives, a *regularization* factor is typically used.

To improve latency and privacy, state-of-the-art slightly optimizes the slicing and finds a more balanced point where the joint privacy latency is optimized. However, to the best of our knowledge, *there is no existing work that investigates this further*. This is where we introduce our two **main contributions** of this paper: adversary-aware and latency-aware designs to improve latency and privacy in state-of-the-art (these are steps 3-5 in Fig. 2).

D. Adversary-Aware Design

Before describing the details, it is important to mention that privacy (adversary-aware) and latency (latency-aware) features are NOT independent of each other, and adjusting one impacts the other. As a result, both metrics should be optimized *jointly*. However, to simplify the description, we first assume that the network is sliced while considering latency, and now we are focusing on privacy (i.e., adversary-aware). In Section III-E, we then describe how both can be jointly optimized.

More formally, we formulate our problem using the following definitions.

***Definition** 1 (ADV):* The ADVANTAGE (Adv) of a classification algorithm, $f : X \rightarrow Y$ is defined as

$$Adv[f(x)] = \sum_{c=1}^{M} (\boldsymbol{y}(x)_c f(x)_c - (1 - \boldsymbol{y}(x)_c) f(x)_c),$$

where $\boldsymbol{y}(x)_c$ is a binary vector indicating the ground-truth label for the input x, and f outputs the likelihood ($0 \leq p \leq 1$) for each class (of total M classes). Given this definition, for each observation, we have $0 \leq Adv(f) \leq 1$. Using this definition, we then define *Privacy Leakage* as follows:

***Definition** 2 (PM):* To measure privacy leakage, we define PRIVACY METRIC (PM) as:

$$PM(\boldsymbol{A}) = \frac{\sum_{\forall x \in \mathcal{X}} Adv(\boldsymbol{A}(x))}{n},$$

where $\mathcal{X} = \{X_1, X_2, ..., X_n\} \in \mathbb{Z}^{w \times h \times n}$ is an input dataset and $\boldsymbol{A}(.; \theta_A)$ is the model used by the adversary to infer the predefined private attribute.

***Definition** 3 (Junction and Cut):* To train an encoding algorithm for a neural network, an L layer network is CUT into *two* pieces, *encoder*, E, and *classifier*, F. The first d layers are assigned to E, while the rest, $L - d$, are assigned to F. In existing algorithms, the attacker receives its input from the JUNCTION of the cut, i.e., the output of the layer d.

In this setting, an adversary could leverage this fundamental vulnerability where during the training only a ***pre-defined proxy*** for the attacker is assumed (called *discriminator or* D), while during the inference, the adversary can *adaptively* change their strategy. Particularly, an adversary has two major *control knobs* to extract information: *location* and *architecture*.

***Definition** 4 (Adaptive Adversary):* To launch an attack, an adversary can perform either or both of these methods:

$$\text{(ARCH.)} \quad Find\, \boldsymbol{A} \;\; st. \quad PM(\boldsymbol{A}) \geq PM(\boldsymbol{D}),$$
$$\text{(LOC.)}\; Find\, \boldsymbol{l} \;\; st. \;\; [(l > d) \&\& [PM(\boldsymbol{D}(x_l)) \geq PM(\boldsymbol{D}(x_d))]].$$

The first strategy indicates that an adversary can find a new architecture such that it can achieve higher information leakage compared to the discriminator architecture used during the training. The second strategy suggests that an adversary can find a new location, l, that is different from the original location of D during training (noted as d) and can achieve higher leakage. Note that this new location should be in later layers of the network (i.e., $l > d$) since the existing methods [6], [10], [25], [45]–[51] all assumed that the discriminator is connected to the JUNCTION (cf. Definiton 3).

The ultimate goal in designing an encoding algorithm is to find the best tradeoff between privacy and accuracy. While there are a few studies for finding a theoretical lower bound for the privacy-accuracy trade-off [10], [47], [52], [53], a practical consideration is if and how the optimum trade-off can be found? To explain, we draw analogies from cryptography where the objective is fairly similar.

***Definition** 5 (Perfect Encoding):* An encoding algorithm, $R(E, D, F)$, is PERFECT if $\forall m_i \in M$ and $c_i = E(m_i; \theta_E)$ and any "efficient" algorithm, $A(c_{j|\, i \leq j \leq L}) = s_c$, we have:

$$Pr[A(c_i) - \frac{1}{|S|}] \leq \epsilon,$$

where ϵ is 'negligible' (user-defined), $S = \{s_1, s_2, ..., s_l\}$ is a discrete set of l private labels, and L is the total number of locations. Perfect encoding can also be formulated by Definition 1, where the goal is to achieve $Adv(A) = r \overset{R}{\leftarrow} S$ (i.e., r is chosen at random from set S).

Given the definitions above, the fundamental challenge is computing $Adv(A)$ for any arbitrary algorithm given an adaptive adversary. Unlike cryptosystems, finding the best-known attack strategy for machine learning is not well-formulated, as the field is continuously evolving [54]. Instead, we propose a ***heuristic optimization algorithm*** where instead of finding the unknown theoretical limit, one can perform an *iterative search* in the state space domain and pick the best candidate among the data points.

Specifically, we propose a *metaheuristic evolutionary algorithm* based on the method proposed by Real *et al.* [55]. The goal is to find *a more sophisticated adversary* so that the encoder becomes more robust and hence the end-to-end privacy increases.

Unlike existing algorithms, the problem we have in Chery is slightly different since the adversary does not have a *fixed* input (location strategy in Definition 4). Noting this difference, the evolutionary algorithm designed in Chery has two main parameters to consider: *input* and *model*. In existing models, including the algorithm developed by Real *et al.* [55], the assumption is that the input is fixed. In our setup, however, this process should be repeated M times, and only the *best* candidate among all should be chosen. M refers to the number of layers in the classifier. For example, if the original network has $L = 20$ layers, and we split the network to $N = 4$ for the encoder and $M = L - N$, then the search algorithm should find the best network among sixteen possible locations.

Given this new requirement, the execution time of our search algorithm becomes a primary concern since searching one layer (with fixed inputs) is fairly time-consuming as shown in the literature [55], [56], and now *multiple* iterations of that search are needed.

We make *two main optimizations* in Chery to address this issue. First, our initial investigation of a few datasets shows that the final few layers of the classifier consistently perform poorly compared to the initial and intermediate layers. Intuitively, this was expected as the features in the final layers are already heavily filtered and typically only relevant to the main classification task (i.e., the non-sensitive information). The initial and intermediate layers, however, still have relevant information which potentially could be extracted. Using this observation, we adjust Chery to discard the last 25% of the network (this number is based on initial observation on our chosen dataset and may not be always generalizable. We believe as a rule of thumb a number between 20-30% is a safe assumption without sacrificing optimality).

The second optimization proposed is that we observe that some locations consistently outperform the rest. We study this further and find that this phenomenon can be explained by a new metric called prediction probability.

Definition *6 (Prediction Probability):* We define PREDICTION PROBABILITY (PP) as:

$$PP = MI(z_l; s) \times SNR_\theta(z_l, s),$$

where z_l is the input to the layer l and MI is the mutual information between z_l and private task, s. SNR is the ratio between the number of features that are 'important' for the private task and the total number of features in that layer and derived from calculating the importance of each feature for a given task and then selecting the top-n features based on a threshold (θ).

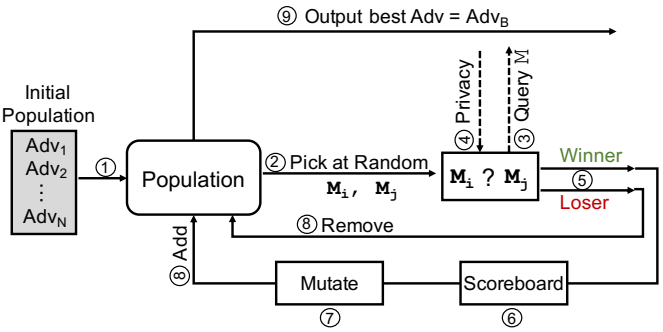


Fig. 3: Details of our heuristic evolutionary algorithm for finding a better, more sophisticated adversary network to improve overall privacy.

Intuitively, PP predicts the layer where an adversary is most likely to infer sensitive features, balancing information vs. noise. Using this, we set $argmax(PP)$ as the location for the adversary's network in our search algorithm.

Using these two optimizations reduces the overhead to only about 1.6-1.65x, as opposed to more than 10x without them (i.e., about an order of magnitude). For the search itself, we use an evolutionary algorithm to find the best-performing adversary. Since in this work, we only focus on image datasets, we only consider CNNs.

The overview of the algorithm is shown in Fig. 3. The algorithm starts with an *initial population*. To make our search more accurate and faster, we observe that creating a manual but carefully-crafted initial population is extremely helpful. To create these candidates, we pick six manually-created networks. Details are provided in Section IV. One key challenge in our setup is that given the input location of the adversary varies, the input size and the structure of each network have to be adjusted for each layer. Therefore, we create multiple instances of the same initial candidate networks with modified sizes – i.e., for the initial population, $P = 6$, we create $P \times M$ networks. Recall that we discard the final quarter of the original network, thus M here refers to $M = .75L - N$.

We then train these hand-crafted groups of networks which then collectively create our *initial population*. At each step of our evolutionary search algorithm, Chery then picks two candidates at random and compare them. The *fitness function* used for comparison is the *accuracy* of the network with regard to inferring private labels. For example, if we have a dataset with human faces, private labels can be defined as inferring people's genders while other attributes in the picture are considered public. The accuracy of the adversary is then defined as how well the candidate network can detect *genders* given an *encoded* input. Details of the datasets and training are provided in Section IV. After comparison, the candidate with higher accuracy stays, and the other candidate is removed from the population by Chery.

Each *winner* candidate after each comparison is then gone through the *mutation* process. We use the strategy proposed by

Algorithm 1: Adaptive Adversary algorithm in Chery.

Data: input x, public task y, private task s, latency t, minimum threshold T, Encoding algorithm $Enc(\theta)$, Max number of iterations TI;
Result: A;
// Step 1: Initialization
Create an initial population manually ($N = P$);
Repeat to create different sizes ($N = M \times P$);
$Loc = argmax(PP)$;
Train $E, F, and\ D$ using data above;
$iter = 1$;
// Step 2: Optimization
while $iter > TI \ || \ (Quality \leq T)$ **do**
  Pick M_i and M_j at random from N;
  Calculate PP_θ using $E_\theta, F_\theta, D_\theta$; // $\theta = \{i, j\}$
  $D_{location} = ArgMax(PP_\theta)$;
  $\{Acc, Pr\} = Enc_\theta(x, y, s, E, D, F)$;
  Mutate $winner(M_i, M_j)$;
  Update N and $Quality$;
  $iter = iter + 1$;
end
Output A;

Real *et al.* [55] where we use a collection of different mutation strategies to create a new *child*. Briefly, the mutation includes adding/removing an entire layer, changing hyperparameters such as learning rate, changing filter size and stride, etc. We refer the readers to the work by Real *et al.* [55] for full details. This newly trained child is then put back into the *population* and the algorithm continues.

The termination criteria for our search algorithm is when Chery reaches a pre-defined iteration OR if the changes in accuracy's improvement are less than a pre-defined threshold for T consecutive comparisons. For the latter, Chery uses a variable called `quality` to keep track of the last comparison and the winner's accuracy. It then uses a counter to terminate the algorithm if needed. Algorithm 1 shows the full details.

The final outcome of the *adversary-aware* step is a network, A, that indicates the best adversary's network and its location. Next, we describe our *latency-aware* strategy and how these two methods are combined and added to our baseline pipeline.

E. Latency-Aware Design

The design of Chery, so far, includes an original network that is sliced into the *baseline* structure with an encoder and a classifier. In Section III-D, we improved this design by designing an evolutionary algorithm that finds a better adversary network for a given slice.

We now turn our focus to finding a better strategy for *slicing*. Combining the slicing, training, and adversary's network, Chery provides notable improvements in privacy and latency over the state-of-the-art.

To find a better slicing, we need to consider two competing phenomena. First, larger encoders are preferred since *(a)* due to the data processing inequality principle [23], less information is available at later layers, and *(b)* more layers assigned to the encoder allow us to train a more powerful encoder that learns what features to remove and what to contain.

The competing factor, however, is the latency constraint. Since the encoder has to be operating on the IoT device, adding a more complex encoder incurs latency and storage overheads. Given the significant imbalance between the computation capabilities of the IoT and the server in a hybrid ecosystem, to minimize latency (and energy consumption) it is preferred to offload as much as possible to the server, hence the smallest possible encoder is desired in a hybrid ecosystem.

Given these two contradicting considerations, Chery uses the following strategy. We make this key observation that instead of adding more layers to the encoder, Chery can employ *perturbation* in addition to encoding which means that the encoder size could be kept unchanged while more privacy is achieved via perturbation. The fundamental difference between *perturbation* and *encoding* is that creating perturbation is significantly less computationally intensive especially if they are created dynamically on the fly or sampled from a pre-trained distribution [57].

While perturbation is computationally cheaper than encoding, naive usage of perturbation could lead to a large reduction in the classifier's accuracy. Note that the key difference between encoding and perturbation is that the encoder intelligently removes sensitive information while perturbation equally impacts both sensitive and non-sensitive information.

Using the above insights, Chery slices the network such that the encoder is the largest possible without violating a pre-defined latency requirement (recall that adding more layers to the encoder means more computation for the IoT and hence higher end-to-end latency). Using this slicing, Chery then starts an iterative process to train the encoder, classifier, and adversary networks.

There are various perturbation techniques in the literature for creating efficient perturbation in a privacy-preserving setting [11], [31], [57], [58]. Chery leverages a perturbation strategy based on the method proposed by Miresghallah *et al.* [59]. Unlike this method, however, Chery uses a combination of encoding and perturbation. This requires generating and updating noise tensors based on the new encoder design after each iteration. Furthermore, unlike the method proposed by Miresghallah *et al.* [59] which leverages mutual information (with a proxy), Chery uses the feedback from the adversary network to train and improve the noise generation block. As extensively discussed in Section III-D, this results in a more accurate estimation of privacy.

The complete overview of Chery's training algorithm with baseline, the adversary-, and latency-aware components are shown in Fig. 4. Briefly, Chery first slices the network with the largest possible encoder without violating the latency requirement. An *initial population* is then created for adversaries using this slicing. Using either adversarial and/or mutual information (cf. Section III-A), the slices are trained. For training, five adversary networks are chosen at random from the initial population. This includes training the encoder, the perturbation, and the classifier.

The trained slices are then fed into the algorithm described in Section III-D, to find the best adversary. The chosen net-

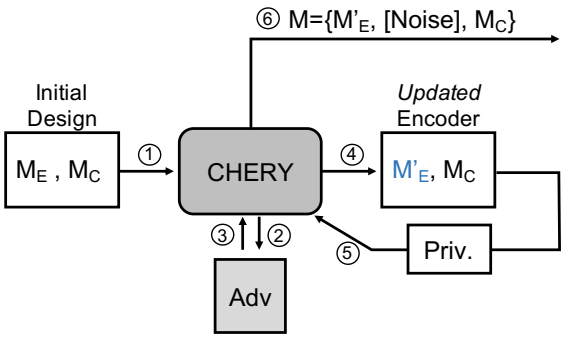


Fig. 4: The steps needed for retraining the network and updating the encoder and perturbation (noise) blocks.

work is then used to *train* the encoder first and the perturbation block next. The final hybrid network is ready at this point.

An important observation made in our design process is the impact of the initial setting (especially the initial population) on the overall quality of the design. To mitigate this dependency, an additional optimization is to rerun Chery' training algorithm using a *different* initial condition and then pick the *best* candidate among all runs. This, however, comes at an additional training overhead.

IV. Implementation and Setup

We use two datasets to evaluate Chery: CIFAR100 and CelebA. They both have been used extensively in the literature. Similar to prior work [24], we use VGG-16 for CelebA dataset and Resnet-18 for CIFAR100. We specifically used two different networks to investigate the applicability of Chery to different networks.

For CIFAR100 we define *fine* classes as private while identifying the *coarse* classes (superclasses) as the main (public) task. Among different superclasses, we particularly focus on "People" and define sensitive information as the fine labels within this class while everything else is considered public.

For CelebA, we define *gender* (*male* attribute) as private information. CelebA has 40 binary features, and we assume the main task is identifying all features except *gender*.

The baseline models are trained on a server with A5000 GPUs and more than 100 GB of memory. We use PyTorch with ADAM optimizer and 0.01 learning rate to train the baseline models. The training and tasks needed for Chery are also implemented and executed on the same server. Using this infrastructure, the Chery's training time for each network was about 8 hours. In Section VI, we discuss various techniques for improving training time.

We use a Raspberry Pi (Model 4) to represent the IoT device for latency estimations in the paper. As described in Section III, Chery slices the network into an encoder and a classifier where the encoder is placed in the IoT device while the classifier is offloaded to the server. When we report latency, we always mean *end-to-end* latency (i.e., encoder, network, and classifier). We assume a Wi-Fi connection between the IoT and Server with a few MB/s data rate.

We use the method proposed by Jaiswal *et al.* called Invariant Representations through Adversarial Forgetting (AdvF for short) [25] as our baseline since it is publicly available and outperforms prior work. To create our initial population described in Section III-D, we use the adversary networks proposed in AdvF [25] and DeepObfuscator [24] and create five variants of those by manually adding one or two more layers and/or changing the hyperparameters.

For our evolutionary-based search, we set $N = 50$, $P = 6$, and $T = 1000$. We observe that our search terminates due to T and not because it reaches the maximum number of iterations.

Metrics. When we report ***accuracy*** in this section, we always report *average* accuracy across all labels. Both networks are first trained with the original network (without encoding) to achieve high accuracy. Similar to prior work, out of more than 200k samples for CelebA, we use 160K images to train, and the rest for testing and validations. For CIFAR-100, 40K samples are used for training, while the other 20K is used for testing and validations.

We report ***privacy*** as the classification accuracy of the private label (i.e., gender for CelebA, and fine classes for CIFAR100) using the attacker's network (A in Algorithm 1). Unlike accuracy, we report the *best* classification accuracy for privacy (i.e., the worst-case).

Finally, the ***latency*** is reported as the *end-to-end* latency starting from the IoT device until the classification is concluded on the server.

V. Results

In this section, we report the results for our adversary-aware (Section III-D) and latency-aware (Section III-E) features. We then report the overall improvement when both are combined (i.e., our context-aware solution).

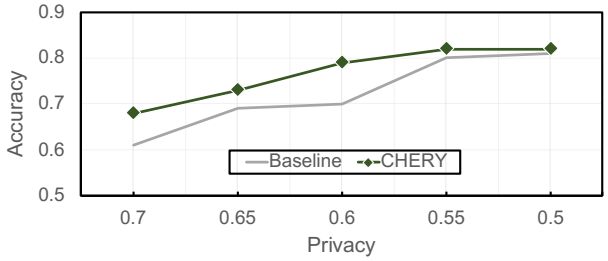
Adversary-Aware Results. We report the privacy-accuracy tradeoff when Algorithm 1 is used to improve privacy. As explained in the previous section, two networks (Resnet-18 and VGG-16) and two datasets (CelebA and CIFAR100) are used in our experiments.

The results are shown in Fig. 5. The "baseline" refers to the original model when Adversarial Forgetting (AdvF) [25] is used, while "CHERY" refers to our model when Algorithm 1 is used to improve the adversary modeling in the baseline model. The figure shows five data points representing different values of T (quality in line 6 of Algorithm 1).

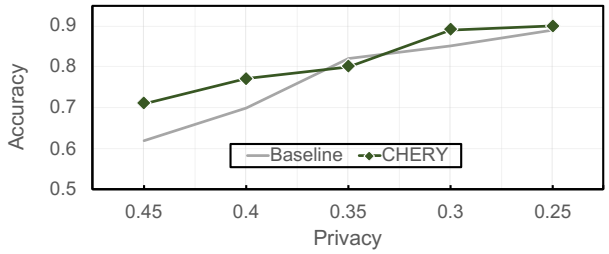
Both optimum accuracy and privacy are sought-after (i.e., the top-left corner of Fig. 5), however, the practical reasons discussed extensively in this paper illustrate that increased privacy adversely affects accuracy, as presented in the figure (i.e., higher privacy results in lower accuracy). As a result, the user has to pick a design point based on their needs.

Our method enhances the baseline model, as demonstrated by the fact that it outperforms the baseline at nearly all design points. This implies that ***Chery grants the user a stronger accuracy-privacy tradeoff*** by either giving improved privacy for the same level of accuracy or vice versa. This is achieved by modeling the adversary more rigorously as described in Section III-D.

The trend is fairly similar for both datasets and among various tradeoff points. The privacy results for CelebA are

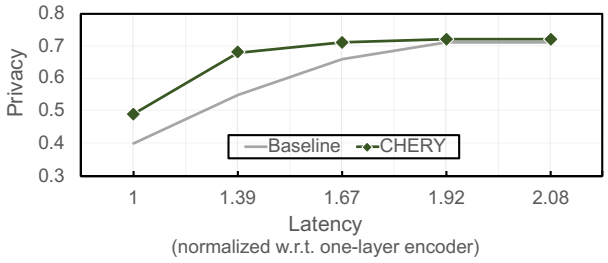


(a) CIFAR-100 Adversary-Aware.

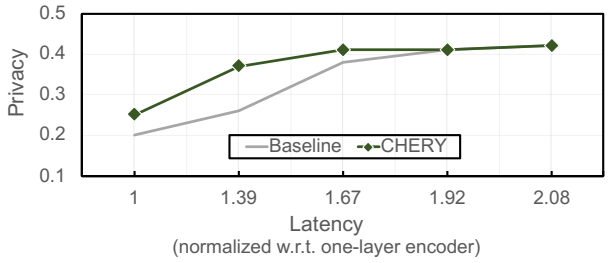


(b) CelebA Adversary-Aware.

Fig. 5: The privacy-accuracy when using Chery's evolutionary algorithm to find a better adversary and retrain the network. The top left corner is the desired spot (higher accuracy and higher privacy).



(a) CIFAR-100 Latency-Aware.



(b) CelebA Latency-Aware.

Fig. 6: The privacy-latency tradeoff for baseline and Chery. Each vertical line is one configuration in our setup. Latency numbers are normalized with respect to the first configuration.

lower since the minimum privacy (random guessing) for CIFAR-100 is $1 - 1/5 = .8$ so at best 80% privacy can be achieved, whereas for CelebA this number is $1 - 1/2 = .5$.

Concerning accuracy, CelebA has a slightly higher accuracy rate because its baseline privacy level is higher than CIFAR-100, around 90% versus around 80%.

An important note to highlight is that the methods proposed by Chery is **orthogonal** to the existing baseline, and hence, can be directly applicable to any new or existing hybrid-based privacy-preserving approach including that proposed recently by Osia *et al.* [23] and Li *et al.* [24].

Latency-Aware Results. We report the latency-privacy tradeoff when the method described in Section III-E is used. Similar to the adversary-aware experiment, we compare the result with the same baseline model (AdvF on Resnet-18 and VGG-16).

Results are shown in Fig. 6. Similar to the previous experiment, we analyze the impact of Chery on both datasets/networks. Unlike the previous experiment, the x-axis here shows the latency, i.e., each vertical line refers to one slice (same encoder size) and hence similar latency. Numbers are normalized with respect to a setup with only one layer as the encoder. The y-axis shows privacy for a given slice in each setup. Note that the latency is reported as *end-to-end* latency which includes latency for computing the encoder on the IoT device (Raspberry Pi), network, and server.

The results indicate that for a given slice (i.e., same size encoder), Chery achieves better privacy. Alternatively, for a given privacy budget, Chery improves the overall latency.

Regarding the privacy-accuracy tradeoff, we present the results for CIFAR-100 dataset in Fig. 7. The results are

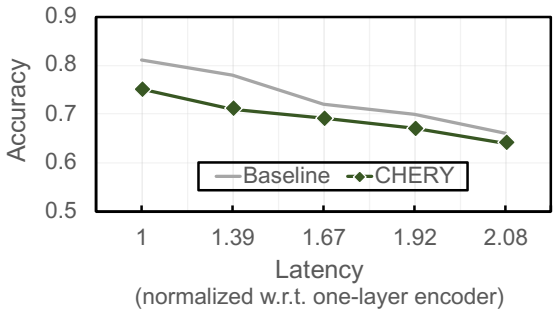


Fig. 7: Decrease in accuracy when using noise+encoding (Chery) vs. when only using the encoding (baseline).

shown for the same setup described in Fig. 6. The important observation is that adding noise has an impact on accuracy. This impact, however, is mitigated as the encoder becomes more sophisticated with more layers as Chery needs to do *less work* to achieve the desired level of privacy.

For a shallower encoder, however, Chery needs to add more noise which results in a larger drop in accuracy. While not shown here, we observe a similar trend for CelebA dataset.

Context-Aware Results. To conclude the results, we examine the impact of using Chery compared to the baseline when *both* features are used. To measure this, we combine the steps described in the first and second experiments. The complete step-by-step design can also be seen in Fig. 2 (cf. Section III).

The results are shown in Fig. 8 for CIFAR-100 dataset. In this figure, the x-axis shows the encoder size (as a proxy to latency), whereas the first bar shows the *original* configuration

	Chery (*this work*)	AdvF [25]	Perturbation [57]	DeepObf [24]	Hybrid [23]
Modeling the Adversary	●	○	○	◐	◐
Considering Latency	●	○	◐	◐	◐
Training Difficulty	◐	●	●	◐	●
Privacy	●	◐	◐	●	◐
Dynamic Perturbation	●	○	●	○	●
Accuracy	●	●	◐	●	◐

TABLE I: Comparison to the prior work. For each category ● means that best in that category (e.g., for privacy higher is better but for training difficulty ● means less difficulty/complexity).

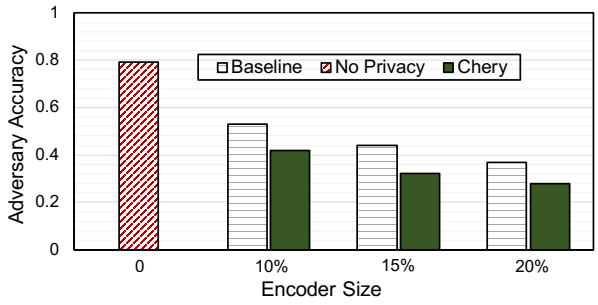


Fig. 8: Adversary accuracy (lower is better) when using Chery with adversary- and latency-aware features compared to the baseline and the original configuration with no privacy.

where no encoding is used. The y-axis shows the adversary accuracy (i.e., $1 - privacy$). For the original network, this is around 79% as this essentially becomes the accuracy of the original classifier to infer *fine* classes.

The other three groups of bars each show one configuration for the encoder where the percentage indicates the amount of computation assigned to the encoder (effectively the slice location). The results are shown for the baseline and Chery when BOTH adversary- and latency-aware features are used.

The results reveal that Chery improves the state-of-the-art by around 10% on average across all encoder sizes, where this gain is achieved through the combination of adversary- and latency-aware features (similar results were observed for CelebA, although not shown here for brevity).

An interesting observation is that the results in Fig. 8 are not the linear combination of what has been seen in experiments one and two. This was expected as the impact of latency-aware on the baseline is more significant on the baseline compared to the adversary-aware configuration since the encoder has been already improved on the latter.

We did not investigate larger encoders as (a) the latency overhead becomes prohibitive and (b) the loss in accuracy is sufficiently low for encoder sizes 20% and larger.

VI. Discussions

Comparison with prior work. We compare our results in six different categories with four prior works that are most relevant to Chery. The works we compare are Adversarial Forgetting (AdvF [25]), an adaptive perturbation method proposed by Miresghallah *et al.* [57], DeepObfuscator [24], and a hybrid method proposed by Osia *et al.* [23]. The rationale behind choosing these works is that they leveraged encoding to achieve privacy for a deep learning application.

This comparison is shown in Table I. We qualitatively compare our work in categories such as how the adversary is modeled, how each method considers IoT latency, and how difficult and time-consuming it is to train each model. Further, we also compare our model with prior work when considering privacy and accuracy as the main two factors in any privacy-preserving design as well as whether each method is using a dynamic perturbation mechanism in their system.

To summarize, Chery improves the state-of-the-art in various factors including latency, privacy, and accuracy. This comes thanks to the more accurate modeling of the adversary and considering latency and privacy jointly.

Training Time. One key challenge in Chery is the increase in training time due to adding adversary- and latency-aware features. Leveraging a heuristic search specifically can increase the training time and difficulty, especially for larger networks.

To address this issue, several optimization techniques at the algorithm and hardware levels could be employed. For example, as discussed in Section III-D, by discarding some portion of the network, the search time for the evolutionary search algorithm could significantly decrease.

Further, as described in prior work [55], hardware parallelization could greatly improve the training time. Given that the candidates in the search algorithm can be analyzed independently, multiple *worker* threads could perform the search in parallel. Finally, the search size itself could be adjusted as the execution time is essentially a tradeoff between training time and the quality of the network.

For this work, as discussed in Section IV, only a few hours were required to perform the training. For larger networks, however, the above optimization techniques could be used more aggressively to balance the training time with the design quality. Lastly, the training time itself is not a huge concern as this needs to be done only once for each network.

Scalability. Another important consideration for Chery, and generally other encoding and/or perturbation-based privacy-preserving mechanisms is scalability. Existing work mostly focused on simple to medium size neural networks. However, the current trend in utilizing highly deep neural networks raises the scalability question for these methods.

The fundamental challenge in using encoding for a larger network is the overhead incurred by the IoT device due to the need for a larger encoder, which results in greater latency. Consequently, concerns about latency become even more pressing, and techniques such as Chery that can intelligently balance between privacy and latency become more useful.

The other orthogonal problem here to address scalability is designing more sophisticated adversarial training algorithms. A better baseline could improve the overall privacy and latency. As mentioned earlier, the key advantage of Chery is

that it is applicable to any adversarial-based method and hence could directly get benefit from a better algorithm.

Metric for Privacy. Finding an effective strategy for measuring privacy is an important and critical challenge in our setup and other related work. Some work used similarity metrics to measure privacy [24]. Others borrowed the concept of k-anonymity [23]. Another group of work leverages mutual information as the main indicator [31]. However, most works have used a *proxy* network to measure privacy.

The underlying rationale for using a neural network to estimate privacy is that such a setup is closest to reality and potentially would be what an actual adversary use to extract information. The recent advancements of theory and implementation in image classification and computer vision have further strengthened this theory given that machines largely outperform humans in most image classification datasets.

Privacy-Preserving Capabilities. Chery provides a context-aware privacy-preserving scheme which improve the state-of-the-art in two fronts. First, for the same latency budget, the overall privacy is improved since Chery can model the adversary more efficiently and/or employs perturbation to satisfy the privacy requirements without sacrificing latency. Second, for a given privacy budget, Chery improves the overall latency and accuracy. This is achieved by the latency-aware strategy explained in Section III and the adaptive algorithm described in the same section.

VII. Conclusions

A novel privacy-preserving technique using encoding and perturbation was proposed for improved privacy. Two main contributions were made to this approach. Firstly, an evolutionary algorithm-based method was proposed to systematically assess the robustness of the encoding technique against potential adversaries. Secondly, a dynamic obfuscation strategy was developed to balance latency requirements and privacy demands in a realistic IoT-cloud hybrid setting. The effectiveness of this approach was evaluated against two deep neural networks, measuring important metrics such as accuracy and privacy. Results and analysis showed that the proposed method enhances privacy in IoT-cloud ecosystems.

References

[1] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *ASPLOS*, 2017.

[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, 2019.

[3] S. Yao, J. Li, D. Liu, T. Wang, S. Liu, H. Shao, and T. Abdelzaher, "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 2020.

[4] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *The 25th annual international conference on mobile computing and networking*, 2019, pp. 1–16.

[5] S. H. Hashemi, F. Faghri, P. Rausch, and R. H. Campbell, "World of empowered iot users," in *2016 IEEE first international conference on internet-of-things design and implementation (IoTDI)*. IEEE, 2016.

[6] A. Singh, A. Chopra, E. Garza, E. Zhang, P. Vepakomma, V. Sharma, and R. Raskar, "Disco: Dynamic and invariant sensitive channel obfuscation for deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 125–12 135.

[7] S. Liu, J. Du, A. Shrivastava, and L. Zhong, "Privacy adversarial network: representation learning for mobile data privacy," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 4, pp. 1–18, 2019.

[8] B. Baron and M. Musolesi, "Where you go matters: a study on the privacy implications of continuous location tracking," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 4, pp. 1–32, 2020.

[9] L. Velykoivanenko, K. S. Niksirat, N. Zufferey, M. Humbert, K. Huguenin, and M. Cherubini, "Are those steps worth your privacy? fitness-tracker users' perceptions of privacy and utility," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 4, pp. 1–41, 2021.

[10] Z. Wu, Z. Wang, Z. Wang, and H. Jin, "Towards privacy-preserving visual recognition via adversarial training: A pilot study," in *ECCV*, 2018.

[11] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," in *KDD*, 2018.

[12] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *CCS*, 2018.

[13] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (S&P)*, 2017.

[14] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "XONN: XNOR-based oblivious deep neural network inference," in *Proceedings of the 28th USENIX Security Symposium*, 2019.

[15] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[16] I. Mironov, "Rényi differential privacy," in *Proceedings of the 2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 2017.

[17] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *ICLR*, 2019.

[18] F. Mo, A. S. Shamsabadi, K. Katevas, S. Demetriou, I. Leontiadis, A. Cavallaro, and H. Haddadi, "DarkneTZ: Towards model privacy at the edge using trusted execution environments," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2020.

[19] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proceedings of the 27th USENIX Security Symposium*, 2018.

[20] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via MiniONN transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017.

[21] M. Malekzadeh, R. Clegg, and H. Haddadi, "Replacement autoencoder: a privacy-preserving algorithm for sensory data analysis," in *Proceedings-ACM/IEEE International Conference on Internet of Things Design and Implementation, IoTDI*, 2018.

[22] O. Hajihassnai, O. Ardakanian, and H. Khazaei, "Obscurenet: Learning attribute-invariant latent representation for anonymizing sensor data," in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 40–52.

[23] S. A. Osia, A. Shahin Shamsabadi, S. Sajadmanesh, A. Taheri, K. Katevas, H. R. Rabiee, N. D. Lane, and H. Haddadi, "A hybrid deep learning architecture for privacy-preserving mobile analytics," *IEEE Internet of Things Journal*, 2020.

[24] A. Li, J. Guo, H. Yang, F. D. Salim, and Y. Chen, "Deepobfuscator: Obfuscating intermediate representations with privacy-preserving adversarial learning on smartphones," in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 28–39.

[25] A. Jaiswal, D. Moyer, G. Ver Steeg, W. AbdAlmageed, and P. Natarajan, "Invariant representations through adversarial forgetting," in *AAAI*, vol. 34, no. 04, 2020, pp. 4272–4279.

[26] Q. Xie, Z. Dai, Y. Du, E. Hovy, and G. Neubig, "Controllable invariance through adversarial feature learning," *Advances in neural information processing systems*, vol. 30, 2017.

[27] X. Gitiaux and H. Rangwala, "Fair representations by compression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[28] U. Gupta, A. M. Ferber, B. Dilkina, and G. Ver Steeg, "Controllable guarantees for fair outcomes via contrastive information estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 7610–7619.

[29] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow: Secure tensorflow inference," in *Proceedings of the 2020 IEEE Symposium on Security and Privacy (S&P)*, 2020.
[30] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade, "TAPAS: Tricks to accelerate (encrypted) prediction as a service," in *ICML*, 2018.
[31] F. Mireshghallah, M. Taram, P. Ramrakhyani, A. Jalali, D. Tullsen, and H. Esmaeilzadeh, "Shredder: Learning noise distributions to protect inference privacy," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 3–18.
[32] R. Liu, L. Garcia, Z. Liu, B. Ou, and M. Srivastava, "Secdeep: Secure and performant on-device deep learning inference framework for mobile and iot devices," in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 67–79.
[33] Z. Liu, J. Weng, J. Guo, J. Ma, F. Huang, H. Sun, and Y. Cheng, "Pptm: A privacy-preserving trust management scheme for emergency message dissemination in space–air–ground-integrated vehicular networks," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5943–5956, 2021.
[34] Y. Cheng, J. Ma, Z. Liu, Z. Li, Y. Wu, C. Dong, and R. Li, "A privacy-preserving and reputation-based truth discovery framework in mobile crowdsensing," *IEEE Transactions on Dependable and Secure Computing*, 2023.
[35] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International conference on machine learning*. PMLR, 2016, pp. 201–210.
[36] Q. Lou, W.-j. Lu, C. Hong, and L. Jiang, "Falcon: fast spectral inference on encrypted data," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2364–2374, 2020.
[37] Q. Lou, B. Feng, G. Charles Fox, and L. Jiang, "Glyph: Fast and accurately training deep neural networks on encrypted data," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9193–9202, 2020.
[38] A. Brutzkus, R. Gilad-Bachrach, and O. Elisha, "Low latency privacy preserving inference," in *International Conference on Machine Learning*. PMLR, 2019, pp. 812–821.
[39] M. van der Hagen and B. Lucia, "Client-optimized algorithms and acceleration for encrypted compute offloading," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 683–696.
[40] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2505–2522.
[41] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: Secure multi-party computation meets machine learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4961–4973, 2021.
[42] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016.
[43] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive laplace mechanism: Differential privacy preservation in deep learning," in *Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM)*, 2017.
[44] P. Grubbs, M. Lacharité, B. Minaud, and K. G. Paterson, "Learning to reconstruct: Statistical learning theory and encrypted database attacks," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (S&P)*, 2019.
[45] P. C. Roy and V. N. Boddeti, "Mitigating information leakage in image representations: A maximum entropy approach," in *CVPR*, June 2019.
[46] Q. Xie, Z. Dai, Y. Du, E. Hovy, and G. Neubig, "Controllable invariance through adversarial feature learning," in *NIPS*, 2017.
[47] H. Zhao, J. Chi, Y. Tian, and G. J. Gordon, "Trade-offs and Guarantees of Adversarial Representation Learning for Information Obfuscation," *NIPS*, vol. 33, 2020.
[48] D. Madras, E. Creager, T. Pitassi, and R. Zemel, "Learning adversarially fair and transferable representations," in *ICML*, 2018, pp. 3384–3393.
[49] D. Moyer, S. Gao, R. Brekelmans, G. V. Steeg, and A. Galstyan, "Invariant representations without adversarial training," in *NIPS*, 2018.
[50] U. Gupta, A. Ferber, B. Dilkina, and G. Ver Steeg, "Controllable Guarantees for Fair Outcomes via Contrastive Information Estimation," in *AAAI*, vol. 35, 2021, pp. 7610–7619, issue: 9.
[51] X. Gitiaux and H. Rangwala, "Fair Representations by Compression," in *AAAI*, vol. 35, 2021, pp. 11 506–11 515, issue: 13.
[52] Y. Xu, S. Zhao, J. Song, R. Stewart, and S. Ermon, "A Theory of Usable Information under Computational Constraints," in *ICLR*, 2019.
[53] C. Song and V. Shmatikov, "Overlearning Reveals Sensitive Attributes," in *ICLR*, 2019.
[54] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*. PMLR, 2019, pp. 6105–6114.
[55] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*. PMLR, 2017.
[56] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *ECCV*, 2018.
[57] F. Mireshghallah, M. Taram, A. Jalali, A. T. T. Elthakeb, D. Tullsen, and H. Esmaeilzadeh, "Not all features are equal: Discovering essential features for preserving prediction privacy," in *Proceedings of the Web Conference 2021*, 2021, pp. 669–680.
[58] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation*, 2008.
[59] F. Mireshghallah, M. Taram, P. Ramrakhyani, A. Jalali, D. Tullsen, and H. Esmaeilzadeh, "Shredder: Learning noise distributions to protect inference privacy," in *ASPLOS*, 2020.

Hossein Khalili (Graduate Student Member, IEEE) is a Ph.D. student at the Secure Systems and Architecture Lab, Department of Electrical and Computer Engineering, University of California, Los Angeles. Prior to UCLA, he earned his B.Sc. and M.Sc. in Electrical Engineering from Sharif University of Technology, Iran. His research interests are security and privacy in machine learning and IoT systems.

Hao-Jen Chien (Graduate Student Member, IEEE) was an MS-PhD student at the Secure Systems and Architecture Lab, Department of Electrical and Computer Engineering, University of California, Los Angeles. Prior to UCLA, she earned her B.SC. degree in Taiwan. Her research interests are machine learning and privacy-preserving computation.

Amin Hass (Member, IEEE) is the R&D Lead of North America Cybersecurity at Accenture, working research projects in the area of secure Trustworthy AI, Quantum Security, 5G and Industrial Internet, and Cyber Risk Management. He earned his Ph.D. in computer engineering from Texas A&M University. Dr. Hassanzadeh has designed and developed multiple prototypes on privacy-preserving ML, alert correlation in IIoT, proactive risk analysis and remediation, and security event classification.

Nader Sehatbakhsh (Member, IEEE) is an Assistant Professor at the Department of Electrical and Computer Engineering at UCLA. His work is broadly focused on developing secure and private systems and architectures and particularly on computer architecture security, IoT security and privacy, and side channels. Prior to joining UCLA in 2020, he earned his Ph.D. in Computer Science from Georgia Tech in 2020, and his B.Sc. in Electrical Engineering from the University of Tehran in 2013.