DAPter: Preventing User Data Abuse in Deep Learning Inference Services

Hao Wu[1], Xuejin Tian[1], Yuhang Gong[1], Xing Su[1], Minghao Li[1 2], and Fengyuan Xu[1*]

[1] National Key Laboratory for Novel Software Technology, Nanjing University
[2] Cornell University

ABSTRACT

The data abuse issue has risen along with the widespread development of the deep learning inference service (DLIS). Specifically, mobile users worry about their input data being labeled to secretly train new deep learning models that are unrelated to the DLIS they subscribe to. This unique issue, unlike the privacy problem, is about the rights of data owners in the context of deep learning. However, preventing data abuse is demanding when considering the usability and generality in the mobile scenario. In this work, we propose, to our best knowledge, the first data abuse prevention mechanism called DAPter. DAPter is a user-side DLIS-input converter, which removes unnecessary information with respect to the targeted DLIS. The converted input data by DAPter maintains good inference accuracy and is difficult to be labeled manually or automatically for the new model training. DAPter's conversion is empowered by our lightweight generative model trained with a novel loss function to minimize abusable information in the input data. Furthermore, adapting DAPter requires no change in the existing DLIS backend and models. We conduct comprehensive experiments with our DAPter prototype on mobile devices and demonstrate that DAPter can substantially raise the bar of the data abuse difficulty with little impact on the service quality and overhead.

CCS CONCEPTS

• **Security and privacy → Security services**; • **Computing methodologies → Computer vision**; • **Human-centered computing → Ubiquitous and mobile computing**.

KEYWORDS

Data Abuse Prevention, Deep Learning Inference Service, Highly-usable Service

ACM Reference Format:
Hao Wu[1], Xuejin Tian[1], Yuhang Gong[1], Xing Su[1], Minghao Li[1 2], and Fengyuan Xu[1]. 2021. DAPter: Preventing User Data Abuse in Deep Learning Inference Services. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449907

*Corresponding author. Email: fengyuan.xu@nju.edu.cn

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449907

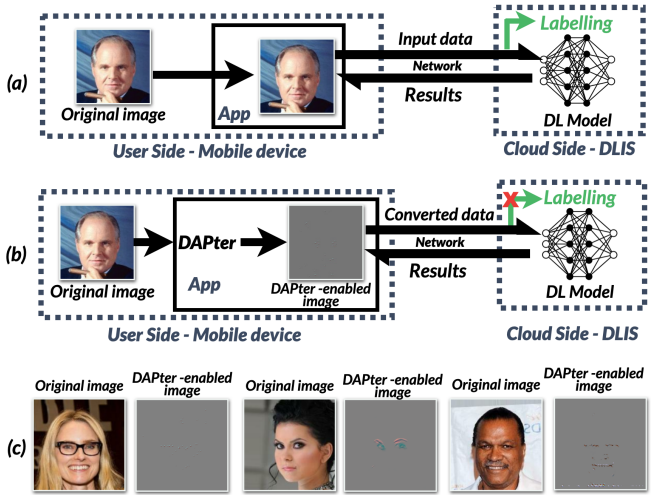

Figure 1: (a) A typical DLIS scenario and the data abuse risk users face. (b) Desired abuse prevention mechanism for DLIS. (c) Sample pairs of original DLIS inputs and their abuse-prevented versions generated by our DAPter.

1 INTRODUCTION

Deep Learning Inference Service (DLIS) [28] lately gains a lot of popularity among mobile end users. It delivers the power of artificial intelligence (AI) to resource-limited devices by offloading expensive computations to the cloud (Figure 1(a)). For example, a mobile end user can enjoy with the support of DLIS the same sophisticated face detection model as the one used on a powerful GPU. Leading web service providers, such as Microsoft, Google, Amazon, and Face++, have achieved great business successes with the DLISes launched for their customers. We envision that the market of DLIS will continue growing at a fast pace, provided that more and more new models are being trained [4].

However, a unique issue has been raised with the rapid development of DLIS. In the context of deep learning (DL), real-world data are always precious assets because they are the key ingredient in training new DL models. Thus, greedy web service providers may collect and exploit the inputs of their DLISes, which are excellent sources of real-world data, to train new models under the table. In fact, some of such providers have already been caught because they explicitly ask users to give up all rights of their input data in the service agreement [2, 5]. End users have been worried about this issue of **data abuse** - *whether their inputs will be stealthily exploited to train other models unrelated to the DLIS they subscribe.*

The data abuse issue in DLIS is different from existing security problems related to privacy. What end users desire is not to remove beforehand all sensitive information in their inputs. Some sensitive information is even commonly used in DLISes, such as facial information and vehicle license plates. However, end users do not want web service providers to freely (at no cost or consensus) utilize their data, no matter whether these data are sensitive or not, in a way they do not pay for (Figure 1(b)). For example, it is unfair to users if their input photos to a paid DLIS of emotion recognition are stealthily collected and labeled to train a gender prediction model, which will be launched as another new paid DLIS. Therefore, this data abuse issue is about the rights of data owners in the web service environment, which is also a key focus of recent regulations like GDPR [31].

In this work, we aim to address the data abuse issue from the perspective of benign web service providers. Those providers are willing to deploy some data abuse prevention for their DLISes because of two great benefits. First, the promotion of this prevention feature is able to make their businesses stand out and thus attract more customers who are afraid of being taken advantage of. Second, enabling this prevention feature can also help reduce potential risks of violating existing and future data regulations and antitrust laws. We hope such data abuse prevention could be part of a standard code of practice in DLIS for strengthening the trust between benign providers and their users.

Our solution **DAPter** is *a lightweight DLIS-input converter at the end user side*. DAPter is built by the web service provider for a DLIS and provided as an add-on to their App on mobile devices. Adapting DAPter requires *no* change in the cloud backend and has *little* impact on the online service quality and latency. When deployed, DAPter effectively and efficiently converts original DLIS inputs into their abuse-prevented versions (Figure 1(c)) before sending them to the cloud backend. The new abuse-prevented inputs are able to maintain the similar inference accuracy of their original counterparts, given the same unchanged DLIS model.

DAPter's conversion prevents data abuse by rendering the DLIS inputs unrecognizable to human beings and AI, so that these real-world data cannot be labeled manually or automatically for the new training [1] [3]. In this work, we take an entropy reduction approach to realize such conversion. It is transparent to existing services and suitable in mobile scenarios, compared to other potential approaches. Comparison details are given in Section 8.

The entropy reduction in DAPter is empowered by our generative model tailored for the considered scenario. This generative model is lightweight enough to run on mobile devices in real time. Its neural structure is independent of the model used in its targeted DLIS, so DAPter can be released to the public for verification or distribution without worrying about the intellectual property of services. We also introduce a novel loss function, the *data abuse prevention loss*, to train our model on how to convert images into desired abuse-prevented ones. The objective of this loss is to discover the entropy that contributes little to the high-level features used in the targeted DLIS model, and eliminate it to the extent where conversion outcomes are satisfied. Please note that our training does not update the targeted DLIS model.

We conduct thorough evaluations on our DAPter prototype implemented on the resource-limited devices, including the Snapdragon 855 Plus, Kirin 960, and Helio X30. Experimental results show that, given our practical security model, DAPter is effective in preventing its outputs from being labeled or restored back via possible attacks. Compared to original images, DAPter's abuse-prevented images only downgrade the inference accuracy within 3% and introduce an extra service latency of $109ms \sim 309ms$. Additionally, the network bandwidth usage of DAPter is $2.1\times \sim 41\times$ more efficient than that of the original DLISes.

In a nutshell, we attempt to *realize the principle of least privilege* (**PoLP**) *in the context of emerging DLIS*. As an initial research effort, DAPter exploits mobile deep learning to prune unnecessary information in input data, which is one of the feasible approaches. We hope our work could attract more research efforts in solving this data abuse issue, or more broadly, the data-right problems in the DL scenarios. The contributions of our work are summarized as follows:

- We investigate the data abuse issue in the scenario of DLIS and propose an entropy reduction solution to address it in a lightweight and intelligent way. Adapting our solution can significantly raise the bar of abusing user data, with minimal influence on existing DLISes. We also leverage the open verification from our community to strengthen the trust between honest providers and their users.
- We design a new generative model with a loss function specifically for the considered scenario, empowering our abuse prevention solution. The DAPter model can convert images into their abuse-prevented versions, and its overhead is small on mobile devices. Proposed loss can train this model to well balance the abuse prevention effectiveness and inference service accuracy. No updates are required on the DLIS model.
- We implement our DAPter prototype on representative mobile hardware and evaluate it with comprehensive experiments. We demonstrate that DAPter is effective in preventing data abuse against different possible attacks under our practical security model, and it is efficient in terms of resource consumption, execution time, and deployment effort.

2 PROBLEM OVERVIEW

We consider a popular DLIS scenario, where image data are taken as inputs. As illustrated in Figure 1(a), a DLIS subscriber submits her images via a mobile App to the corresponding cloud backend and receives the inference results back in real time. Subscribers (i.e., users) want their data to be only used in their paid service and for their best interests. More concretely, they feel taken advantage of if their data may be secretly utilized in training new DLIS models. Meanwhile, an honest web service provider is also willing to demonstrate their DLISes are free of data abuse risks.

Security Model. There are two types of providers, greedy and honest. A greedy provider, although not malicious, would like to abuse the user data of his DLIS for his own interests. He labels these data and uses them in the training of a new DLIS model. On

[1] This work focuses on the supervised learning which requires labeled data. Most models in DLIS so far belong to this category. However, our design is general and could be effective in the weakly-supervised and unsupervised learning scenarios.

the other hand, an honest provider seeks to grow her business via the trust building with users. She would like to prove that all data collected in her DLIS will not be secretly exploited, complying with the data regulations and laws.

We want to help the honest provider to achieve their goal. Our security goal (**S**) is to greatly reduce the possibility that the data sent by users are able to be labeled. It consists of three sub-goals: **(S1)** data should *not* be visually recognizable; **(S2)** data should *only* retain features necessary to subscribed DLIS; **(S3)** the security applied on data can *not* be reversed.

We assume that a piece of code can be trusted as long as it is released to the public, e.g., putting the code with full documentation in a public-accessible git repo. It is because, in our community, there are many good volunteers and organizations offering help with the code function verification, such as the white-hats, NGOs, App stores, and law enforcement authorities. This is the same reason why open source is generally considered more secure than closed source. Without loss of generality, we also assume that a DLIS subscribed by a user only contains one DLIS model, although our prevention can support a DLIS to have multiple models.

Objective & **Challenge.** We would like to introduce, only at the user side, a PoLP solution that can meet our security goal above. Our solution can run on the mobile device as part of the DLIS App. Before data are uploaded to the provider, they are processed by the solution so that only necessary information is retained, i.e., eliminating the chance of data abuse at the provider side.

The key challenge in design is to strike a good balance between security and usability. DLIS subscribers are sensitive to service quality and latency, while their mobile devices have limited resources. Moreover, the provider is sensitive to the deployment cost and the convenience of adopting our solution. Therefore, besides security, we want to achieve three sub-goals of high usability (**U**): **(U1)** good inference accuracy is maintained for DLIS; **(U2)** there is no change in the backend, including the model and architecture; **(U3)** execution time on mobile devices is suitable for online services.

Existing Solutions. We investigate several existing data security solutions and find that they all have deficiencies in serving our design goals. The fully homomorphic encryption (FHE) [7, 10, 12] offers the full-fledged confidential computation, which is able to cover our security needs, but its heavy overhead is not affordable in the DLIS scenario. Although the trusted execution environment (TEE) [14, 29, 30] can provide an efficient environment for secure computing, it requires special hardware (the SGX-enabled CPU) and suffers the side-channel attacks. The DL model partitioning (MP) [19, 23] offloads the front portion of a model onto the device so that only high-level features are sent to the backend. However, an adversary could reconstruct the original data from these features, and the provider also has to give up the offloaded model part as their intellectual property. Last, the differential privacy (DP) [32] requires adding the high-intensity noise on inputs, hurting the inference accuracy a lot [21]. In fact, the applicability of DP in our considered scenario is questionable due to the problem formulation difficulty. Please refer to Section 8 for detailed explanations and comparisons.

3 DAPTER DESIGN

In this section, we introduce our data abuse prevention solution, DAPter, which is a lightweight DLIS-input converter at the end user side. We show both the high-level design intuition and overall design description. Details of the proposed loss function are explained in the next section.

3.1 High-level Design

Our strategy for realizing PoLP is to minimize unnecessary pixel-wise entropy in images before giving them to the provider. We consider a piece of pixel-wise entropy unnecessary if it contributes little to the high-level features used in the targeted DLIS. We select the pixel-wise entropy to manipulate because it is a generic feature working for various image-input DLISes. As a low-level feature, the pixel-wise entropy is also much easier to compute than high-level ones.

The key step in our strategy is identifying the pixel-wise entropy unnecessary to a DLIS. We leverage the local linearity of neural structures [15] to assess the importance of each pixel in the image with respect to the targeted DLIS. More concretely, we introduce an entropy-oriented loss function to train a lite neural network, which can capture the sophisticated relationship between pixel-wise entropy and high-level features.

Our lite neural network is a generative model. Given an image input, it generates an image output, which only retains the pixel information closely related to the high-level DLIS features. The neural structure of our model is generic and independent of various DLIS models. Moreover, trained weights of our model do not leak, by design, any information regarding what the high-level features are in the targeted DLIS. Thus, it can be released to the public for purposes like open verification.

DAPter by design need to be trained for each targeted DLIS model. Different DLIS models rely on different features to perform the inference. The information unnecessary to one model could be critical to another model. Thus, the abuse prevention cannot be one-size-fits-all due to the DLIS task variety. However, this training, as part of the one-time DAPter initialization, only takes several hours, according to our empirical study. Given that a DLIS could last for years, this training cost can be amortized and is acceptable.

3.2 Overall Description

DAPter is an intelligent software running on the DLIS subscriber's mobile device. It could be deployed as a standalone App or part of the existing DLIS App. We show the workflow of DAPter in Figure 2. Whether a subscriber deploys DAPter or not has no influence on other subscribers and the corresponding cloud backend. DAPter's core component is the lightweight generative model trained with a novel loss. The trained model can remove irrelevant information in raw RGB images according to the targeted DLIS, while maintaining excellent QoS like the inference accuracy and latency. DAPter ensures that the provider cannot label the processed images or restore them back to original forms.

3.2.1 Generative Model. Our generative model is tailored from the U-Net [24]. It keeps the U-Net's skip connections, but the channel number and convolution layers are modified in the architecture

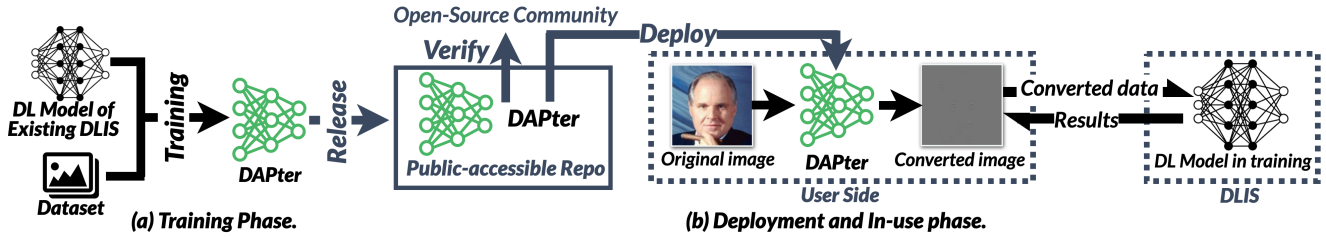

Figure 2: The DAPter workflow.

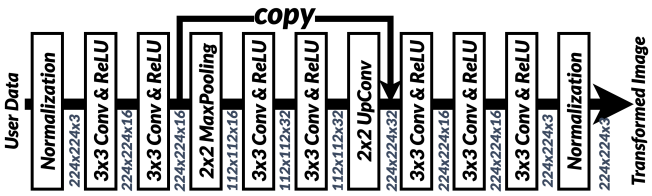

Figure 3: The neural network architecture of DAPter's converter. Assume the input image size is 224 × 224px.

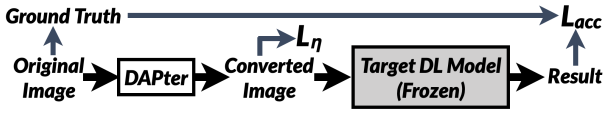

Figure 4: DAPter training Phase.

because our goal is to remove unnecessary semantics rather than keeping them. We present the architecture of DAPter's converter in Figure 3. The converter adopts a symmetrical architecture with the down-sampling and up-sampling modules. This symmetry ensures that the outputs of DAPter are the same size and format as the inputs so that there is no change in the targeted DLIS model. We use a copy connection to forward the output features of the 3rd layer and concatenate them with the input features of the 8th layer. This feature fusion through this "shortcut" can help DAPter well capture the high-level semantic information and low-level spatial information of input data, which is important to the information reduction. Our structure only has eight convolution layers, which is lightweight. When running on the mobile device, our model only adds about $109ms \sim 309ms$ to the end-to-end service latency of a DLIS.

Our model design allows DAPter instances targeted at different DLISes to share the same neural network design, which is DLIS-agnostic. This is beneficial because we could leverage the transferring learning technique [22] to speed up the training. Moreover, our models leak no information about the targeted DLIS model architecture, which is the intellectual property of the web service provider.

Before deployment, our generative model is trained with the targeted DLIS model and, more importantly, our data abuse prevention loss L_{dap} (Equation 1). This loss guides our model in training to learn an effective and efficient way to remove abusable information directly at the pixel level of RGB images, with respect to the targeted DLIS model accuracy. We describe our training procedure below and introduce the data abuse prevention loss in the following section (Section 4).

3.2.2 Training & Deployment. The training architecture is shown in Figure 4. The entropy reduction loss, L_η, is proposed to measure the entropy residual in the converted image (Section 4.1). Minimizing L_η is able to reduce the abusable information available to the backend. Moreover, the converted image is also evaluated in terms of the inference accuracy of the targeted DLIS model. Thus, we introduce the inference accuracy loss L_{acc} (Section 4.2). These two loss functions are applied in an adversarial manner and together form our new loss function called the data abuse prevention loss.

Please note that the targeted DLIS has already been trained by the provider beforehand, and it is fixed during the training of DAPter. At the end of the training, we obtain a DAPter model, which can convert images in a way totally compatible with the original input and QoS needs of the target DLIS. In the meanwhile, converted images can hardly be labeled and exploited to train new DLIS models.

The trained DAPter for its targeted DLIS, together with the necessary running code (both in source and executable formats) at the user side, is released to the public, e.g., in a public-accessible repo. All parts of this release are free to be inspected, verified, tested, and assessed by any party, such as the white-hats, NGOs, App stores, and other law enforcement authorities. As mentioned in our threat model, we assume the model and code, after releasing, are benign because any malicious behavior will be detected eventually. We share the same philosophy as the open-source community - the open-source is relatively secure and trustworthy. Therefore, a user can trust, download, and install the DAPter released for her subscribed DLIS.

Additionally, it is also possible to introduce a trusted third-party to train and endorse DAPter for its targeted DLIS. However, a proper training and releasing procedure is required. We will investigate this deployment manner in the future.

4 DATA ABUSE PREVENTION LOSS

We formulate our data abuse prevention loss function L_{dap} as

$$L_{\text{dap}} = \lambda * L_\eta + (1 - \lambda) * L_{acc} \tag{1}$$

where L_η is the entropy reduction loss, L_{acc} is the inference accuracy loss, and λ is a weighting hyperparameter between 0 and 1. L_η is designed to reduce the pixel-wise entropy retained in the DAPter output, while L_{acc} is designed to maintain the inference accuracy

when the DAPter output is taken as input in the targeted DLIS. Minimizing L_{dap} is the learning objective in the DAPter model training for a DLIS. Details of L_η and L_{acc} are presented below.

4.1 Entropy Reduction Loss L_η

A key step in designing L_η is to measure the pixel-wise entropy in an image. A well-known definition is the image entropy [16], which can quantify the overall pixel-wise information contained in the image. Given a gray-scale image I, it is defined as

$$H_I = -\sum_{i=0}^{255} p_i \log p_i$$

i is a possible pixel value ranging from 0 to 255. p_i is the occurrence possibility of i in I - $p_i = \frac{f_i}{N}$, where f_i is the number of observed i-value pixels in the total N pixels of I. When the image has RGB channels, its entropy is computed as the sum of the entropy of each channel.

However, such pixel-wise entropy calculation is not a trainable metric in terms of the loss function design of DL. Such calculation is a counting process, which is discrete. It cannot be optimized (minimized in our case) by the backpropagation procedure of DL, which computes gradient descents. Therefore, we propose a DL-oriented alternative to replace the image entropy metric as the training loss. Minimizing our loss, i.e., L_η, is equivalent to minimizing the image entropy.

Before introducing the new metric L_η, we first make a supporting statement with our analysis.

Statement 1. *By enlarging the occurrence possibility of a specific pixel value, the upper bound of the image entropy of an image can be reduced.*

Analysis . We assume the occurrence frequency of a specific pixel value j in the image I is known. Then the image entropy can be rewritten as

$$H_I = -p_j \log p_j - \sum_{i \in \{0\sim255\}\setminus\{j\}} p_i \log p_i = H_j + H_{\sim j}$$

The max value of $H_{\sim j}$ will be reached if all pixel values except j has the same occurrence frequency, i.e.,

$$H_{\sim j} \le 255 * \left(-\frac{1-p_j}{255} \log \frac{1-p_j}{255}\right)$$

So the upper bound of H_I is

$$\max H_I = -p_j \log p_j - (1-p_j) \log \frac{1-p_j}{255}$$

Therefore, we have

$$\frac{\mathrm{d} \max H_I}{\mathrm{d} p_j} = \log \frac{1-p_j}{p_j} - \log 255$$

It is obvious that $\frac{\mathrm{d} \max H_I}{\mathrm{d} p_j} < 0$ when $p_j > \frac{1}{256}$. So the larger p_j is than $\frac{1}{256}$, the smaller the upper bound of the image entropy will be. The statement is proved. □

Therefore, according to the above supporting statement, we design an entropy metric η of an image I as

$$\eta(I, I_{\text{ref}}) = |I - I_{\text{ref}}|_1 \tag{2}$$

Figure 5: Training architecture of the auto recognition attack, which is reinforced via transfer learning training.

where I_{ref} is a fixed reference image in which all pixels have the same value, and $|\cdot|_1$ is the L_1 norm operator.

Here we prefer L_1 norm to L_2 norm, which is commonly seen as the distance metric, because of the following reasons. When optimized with L_1 norm, the outcome is close to the reference target, and some pixels of the outcome are exactly the same as the reference image. L_2 norm can make that every pixel is close to the reference image, but probably none of them is exactly the same. Making every image containing similar pixels is critical to DAPter as it means less abusable information is retained in the image, according to our supporting statement. Thus, we use L_1 in η.

Finally, we define our entropy reduction loss L_η as

$$L_\eta = \sum_I \eta(\hat{I}, I_{128}) \tag{3}$$

where $\hat{I}$ is the converted image output of an image input I, and I_{128} is the reference image with each pixel equaling to $(R128, G128, B128)$. Considering that the performance of the neural network is sensitive to the input distribution, we choose the I_{128} to minimize the mean value of the pixel values and the entropy.

4.2 Inference Accuracy Loss L_{acc}

For the sake of simplicity, we adopt the widely-used training loss, the cross entropy as our L_{acc}:

$$L_{\text{acc}} = -\sum_i \text{label}_i \log(\text{prob}_i) \tag{4}$$

where prob_i is the inferred probability of class i (the i-th element of SoftMax output vector of the targeted DL model) for Image I, and label_i is the i-th element of the ground-truth vector whose element corresponding to the ground-truth category of Image I is 1 and rest elements are 0.

5 CONSIDERED ATTACKS

Given our security model, we consider three types of possible attacks against the effectiveness of DAPter. They are auto recognition attack, visual recognition attack, and image reconstruction attack, corresponding to our security requirements. The attack goal of the first two is to directly label abuse-prevented images for new pieces of training. The attack goal of the last one is to restore those images back to original forms and then label them. We introduce these three attacks here and use them in the evaluation of DAPter (Section 7.3).

5.1 Auto Recognition Attack

The AI, specifically the DL, has been leveraged to automatically label data. It often outperforms the human worker in cases like the

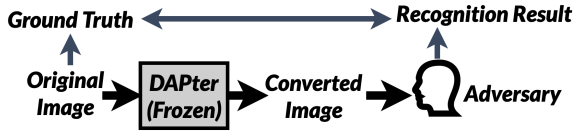

Figure 6: The diagrammatic sketch of the visual recognition attack.

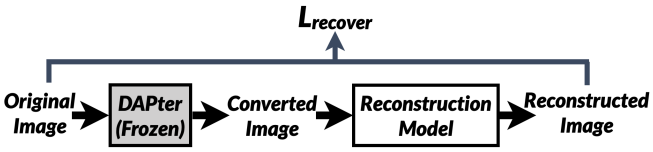

Figure 7: Training architecture of image reconstruction attack. The reconstruction model is a generative model to perform the recovery task.

incomplete data and the blurred data. The adversary can leverage this excellent recognition ability of the DL to label the entropy-reduced outputs of DAPter. We also allow the adversary to reinforce the recognition on DAPter's outputs via transfer learning training. The training procedure is shown in Figure 5. The training loss L_{attack} is the cross-entropy loss (Equation 4) with a regularization part. The training data is prepared in the following way.

The adversary obtains the DAPter corresponding to the targeted DLIS, e.g., the inference of attribute A, from the public repo. He also manually labels a small set of his data with desired ground-truth, e.g., the attribute B. Then, he feeds the data to DAPter to get the abuse-prevented outputs. These outputs with the ground-truth of attribute B are the training data in the transfer learning so that a DL model originally working on normal images can then be fine-tuned to recognize the attribute B in abuse-prevented images.

After the DL model is trained with the transfer learning against DAPter, the adversary will utilize this model to automatically label a lot of abuse-prevented images used in the targeted DLIS (i.e., attribute A inference). The large volume of auto-labeled images is used to train a new DLIS model related to attribute B.

5.2 Visual Recognition Attack

What DAPter generates is still the RGB image, although the entropy in it is reduced. Therefore, the adversary can also find human workers and spend a lot of labor hours to manually label a large amount of data (i.e., the abuse-prevented images) needed in the new DLIS model training. To conduct this attack in our evaluation, we invite volunteers to perform the data labeling work on DAPter's outputs. For example, ask them to label glasses type in abuse-prevented images received by a DLIS inferring the person's age. The attack method is presented in Figure 6.

5.3 Image Reconstruction Attack

As an advanced approach, the adversary can first reconstruct the original image from the abuse-prevented one and then label them manually or automatically. The reconstruction is performed by a generative model trained against DAPter. The training procedure of this attack is shown in Figure 7. This reconstruction model is symmetrical with DAPter, and is trained by optimizing the recovery loss, L_{recover}. It is defined as $L_{\text{recover}} = |I_r - I|_2$, where I_r is the reconstructed image from the abuse-prevented one, and I is the corresponding original image. Please note that DAPter is fixed during the training. To make this attack more powerful, we even allow the adversary in our experiments to utilize the same dataset used in the training of DAPter.

Table 1: Hardware platforms used in the prototype implementation.

Platform	CPU	RAM	OS
Snapdragon 855 Plus	1× 2.96GHz Kryo 485 Gold 3× 2.42GHz Kryo 485 Gold 4× 1.80GHz Kryo 485 Silver	8GB	Android 9
Kirin 960	4× 2.40GHz A73 4× 1.80GHz A53	6GB	Android 9
Helio X30	2× 2.60GHz A73 4× 2.20GHz A53 4× 1.90GHz A35	4GB	Android 9

Table 2: The model type of the DLIS models used in experiments.

Model Name	Structure Info
LeNet [18]	A classical convolutional neural network. It consists of two convolution layers and two fully connected layers.
VGG [27]	The first attempt at the depth of CNN. We adopt the *VGG11*, which has 8 convolutional layers with 3 fully connected layers, and *VGG16*, which has 5 more convolutional layers, in the following experiments.
ResNet [11]	A widely-used DL model, which is remarkable in many tasks. We adopt the *ResNet18* (18 convolutional layers) and *ResNet50* (50 convolutional layers) in the following experiments.

6 EVALUATION SETUP

In this section, we present our evaluation setup, including the DAPter implementation, considered DLISes in use, and corresponding new DLISes if the data abuse occurs. The evaluation results are provided and analyzed in Section 7.

DAPter Implementation. We implement DAPter on the top of the Tensorflow v1.13.1 [6]. We use a PC with four NVIDIA TITAN Xp GPUs for the offline training of DAPter for the targeted DLIS model. The training time, on average, is about 90*mins* regardless of what the targeted DLIS model is. Trained DAPter is wrapped as an Android App with the TensorFlow Inference Interface, and the size of this App is around 1M. In our evaluation, we deploy and test DAPter on three different mobile SoCs, including the Snapdragon 855 Plus, Kirin 960, and Helio X30. Their basic information is listed in Table 1. DAPter's runtime overhead is shown in Section 7.4.

Considered DLISes. The DAPter prototype connects to the service backend through the 4G network. Our backend is a typical machine with Intel Core i7-6850K CPU, 128G memory, and one NVIDIA TITAN Xp GPU, and it hosts various DLISes, which we consider in our evaluation. The models in these DLISes are built from different combinations of three classic DL structures and six representative datasets. Detailed information is provided in Table 2 and Table 3, respectively.

Abuse Settings. The six datasets mentioned above can be categorized into three classes, the single-label dataset (SLD), the multi-label dataset (MLD), and the hierarchical-label dataset (HLD). Each

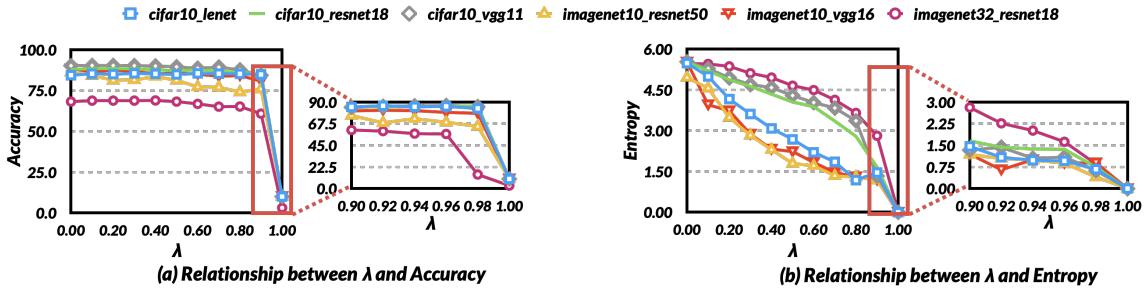

Figure 8: The average task accuracy and entropy of the images produced by converters trained with different λ values. (Section 7.1)

image in SLD has one ground-truth label. Each image in MLD has multiple ground-truth labels that are not necessarily correlated. For instance, a portrait photo has the label "sunglass" and the label "male". In an HLD, although each image only has one label, all labels are correlated and form a hierarchy. For example, an image labeled with "cat" belongs to an aggregated label "animal". Each DLIS built above picks either a single label in an MLD or an aggregated label in an HLD as its inference goal. Therefore, given such DLIS, the rest of MLD labels or sub-labels of used aggregated labels can be utilized as the ground-truth when evaluating DAPter's abuse prevention effectiveness against considered attacks.

7 EVALUATION

In this section, we perform a comprehensive evaluation on our DAPter prototype. First, we illustrate how to determine a suitable hyperparameter λ in Equation 1. Then, we analyze DAPter's performance from the perspective of DL interpretability. In the last two subsections, we demonstrate that DAPter can achieve all our design requirements, i.e., the data abuse prevention **(S)** and the high usability **(U)**, defined in Section 2.

7.1 Hyperparameter λ

Hyperparameter λ, in Equation 1, is used to balance the security and the usability of DAPter converter. A larger λ can let the converter try to remove more pixels when performing the prevention. However, a large λ may also cause too much information being removed from user input and impair its usability, i.e., the accuracy of the targeted task. In the following, we explore how λ affects the trade-off and give an empirical λ value. *The brief conclusion is that $\lambda = 0.9$ is a sweet point to balance security and usability.*

Settings. Here we experiment with two datasets, i.e., Cifar10 and ImageNet10 (Table 3). We use all five DL networks introduced in Table 2 as our considered DLIS models. In what follows, we use a dataset name and network name pair to denote different DLIS tasks: Cifar10-LeNet, Cifar10-ResNet18, Cifar10-VGG11, ImageNet10-Resnet50, ImageNet10-VGG16, and ImageNet32-ResNet18.

We train each DLIS task with λ values ranging from 0 to 1 and present the results in Figure 8. As we can see, a larger λ value results in lower task accuracy and leads to a lower image entropy. When the λ value changes from 0 to 0.9, the task accuracy remains basically unchanged (the average drop is less than 3%), but the entropy value of the converted image drops to less than 30% of what it is before protection. When the λ value gets greater than 0.9, both the task accuracy and the entropy value will decline rapidly. This means that when the λ value is around 0.9, our DAPter can effectively balance the entropy value and task accuracy. Next, we will experimentally prove the relationship between entropy value and protection effectiveness.

Table 3: The datasets used to train the DLIS models and the corresponding DAPter converters in our experiments. SLD is short for single-label dataset, MLD is short for multi-label dataset, and HLD is short for hierarchical-label dataset.

Name	Type	Dataset Info
CelebA [20]	MLD	A large scale face dataset including more than 200,000 celebrity images. Each image has 40 attribute annotations.
LFW [13]	MLD	A face image dataset containing 13,000 images, and each image has labels on identity and gender.
Cifar10 [17]	SLD	A dataset consisting of 60,000 32x32 color images in 10 classes, with 6,000 images per class.
Cifar20:100	HLD	A dataset built on top of the *Cifar100* [17] dataset with 60,000 images in 100 classes. We group the 100 classes of *Cifar100* into 20 aggregated superclasses by semantics. The aggregated 20-superclass dataset is denoted as *Cifar20*. Therefore, *Cifar20:100* shares the same dataset with *Cifar100*, and each image of *Cifar20:100* has an original class label and a superclass label.
ImageNet10	SLD	A dataset consisting of 10 randomly-picked classes from ImageNet [8].
ImageNet8:32	HLD	A dataset built on top of the ILSVRC2012 dataset [25]. ILSVRC2012 is a widely used subset of ImageNet, which contains 1,286,167 images in 1,000 classes. According to the "popular synsets" released on [1], we form an 8-category classification dataset called *ImageNet8* by the first eight synsets belonging to "animal" and "instrumentation". For each of these eight classes, we randomly choose four child synsets and form up a 32-category classification task called *ImageNet32*. Therefore, *ImageNet8* and *ImageNet32* share the same data but support different tasks.

To explore the relationship between entropy value and prevention effectiveness, we perform the image reconstruction attacks (Section 5.3) on the converted images produced by converters trained for different tasks with different λ values. We present some example images in Figure 9(a), and it shows that as the λ value increases, the quality of reconstructed images becomes worse. We measure the quality of recovered images through the structural similarity (SSIM) index in Figure 9(b). It shows that a larger λ value can bring a lower SSIM index value. At the point where $\lambda = 0.9$,

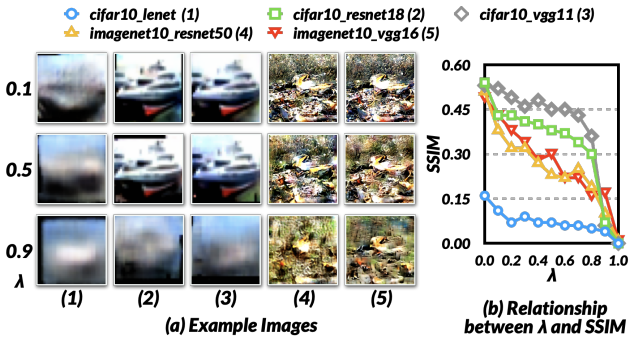

Figure 9: Part of the recovered images and the average recover quality under different settings.

the SSIM is less than 0.2, which means the recovered images have very bad quality [33]. That is, the attackers can hardly perform the image reconstruction attack when applying $\lambda = 0.9$ in the DAPter system. We notice that the reconstruction attack result on LeNet is bad even if λ is set to 0. This is because LeNet is a small network and is under-fitting on the Cifar10 dataset.

We find that $\lambda = 0.9$ is a sweet point in all experiments to balance the prevention effectiveness and the targeted task's accuracy since the task accuracy drops only less than 3% when $\lambda = 0.9$. That is, the converter trained with $\lambda = 0.9$, which removes most information unrelated to the targeted task and preserves the useful information, can achieve high performance on both security and usability. This λ will be set as 0.9 in rest evaluations unless otherwise specified.

λ Tuning Strategy. In a practical scenario, the default λ value 0.9 can be first used by the DLIS provider to train the DAPter instance. Then the provider can measure the security and usability indices, i.e., entropy value and the drop in task accuracy, on the outputs of trained DAPter. If both indices can meet the criterion (lower than 3 and 3% respectively), which is most like to occur according to our empirical study, no further tuning is needed; if the security index is higher than 3, the provider can increase the λ value by 0.02 and perform a fine-tuning on the DAPter instance; if the accuracy index is higher than 3%, the provider can reduce the λ value by 0.02 and perform a fine-tuning. The above step is repeated until both indices meet the criterion.

7.2 Conversion Quality

We analyze the converted results of DAPter via the saliency map. The saliency map is a visualized map interpreting, in which pixels of an image are highly relevant to the inference accuracy of a targeted DL model.

Metrics. We choose the SOTA approach *Grad-CAM* [26] to generate a pixel-wise saliency map. Given an inference input image and a class C, *Grad-CAM* can generate a saliency map representing which part of the input supports the CNN model to decide C. The saliency map has the same size as the user inference input, and it can describe how important each pixel is to the inference result. The pixels with larger importance values contribute to the high accuracy of the targeted DLIS. We compute the importance value V of the user inference input I for the inference task C as: $V = |\text{Grad} - \text{CAM}(M, I, C)|_1$. M is the targeted DLIS task. We also compute the importance value V_D of the user inference input I for the DAPter-enabled inference task C as: $V_D = |\text{Grad} - \text{CAM}(M_D, I, C)|_1$. M_D consists of a DAPter converter and its corresponding DLIS model, which is defined as: $M_D(\bullet) = M(\text{Converter}(\bullet))$. The importance values V and V_D can depict which part of the user input I is meaningful to M and M_D, respectively.

Table 4: The normalized input importance value of different converter and DLIS model combinations. (**A** for arched eyebrow inference, **B** for bangs inference, **E** for eyeglasses inference, and **W** for wearing lipstick inference.)

	Converter A	*Converter* B	*Converter* E	*Converter* W
***DLIS* A**	**1**	7.21×10^{-3}	5.02×10^{-4}	4.46×10^{-1}
***DLIS* B**	1.47×10^{-2}	**1**	7.36×10^{-5}	1.22×10^{-1}
***DLIS* E**	9.61×10^{-2}	7.52×10^{-2}	**1**	2.83×10^{-1}
***DLIS* W**	5.02×10^{-10}	1.11×10^{-10}	0.00	**1**

Attention Consistency. We experiment five times by choosing three attributes recognition from CelebA dataset as the targeted tasks. The selected attributes are relevant to the distinct position of the human face. We denote performing saliency analysis on the vanilla DLIS (M) as before-protection saliency analysis. We denote performing saliency analysis on the DAPter-enabled DLIS (M_D) as the after-protection saliency analysis. We show some of the saliency map examples in Figure 10. We can see that the saliency maps generated by the before- and after-protection have high similarity. That is, DAPter's protection procedure doesn't change the operation logic of the DL model of the targeted DLIS, and the model attention is consistent before and after the protection.

Effectiveness. Here we show that the DAPter converter can do remove the information unrelated to the targeted DLIS task and retain useful information. Recall that each converter is trained with the DL model of a targeted DLIS. We train four converters for arched eyebrow inference, bangs inference, eyeglasses inference, and wearing lipstick inference. Then we attach each converter to four inference models, producing 16 converter and DLIS model combinations. We calculate the importance value for different combinations. If our DAPter converter works, the importance value of the combination consisting of the DLIS model and its corresponding converter should be larger than other combinations containing this DLIS model.

The average importance values of the different converter and targeted model combinations are shown in Table 4. We can see the importance of the DLIS model attached by its corresponding converter is at least 2.24× larger than the DLIS model attached by a random converter. That is, DAPter can effectively prevent the unmatched target model from extracting information from the user input protected by the converter, which corresponds to the targeted DLIS.

7.3 Security Evaluation

7.3.1 Auto Recognition Attack. We introduce the auto recognition attack in Section 5.1. Here we show DAPter can effectively resist this kind of attack.

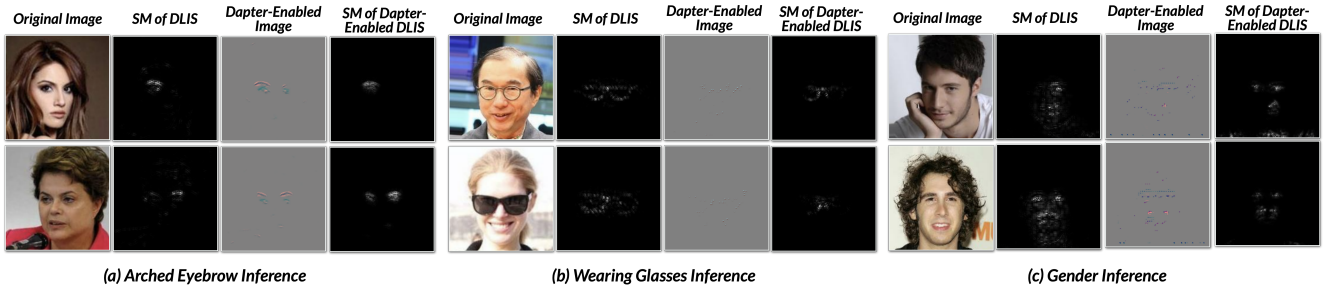

Figure 10: The saliency maps generated by the DAPter-enabled DLIS and the vanilla DLIS. The white part of the saliency map indicates that the pixel at the corresponding position in the original image is of high importance to the results.

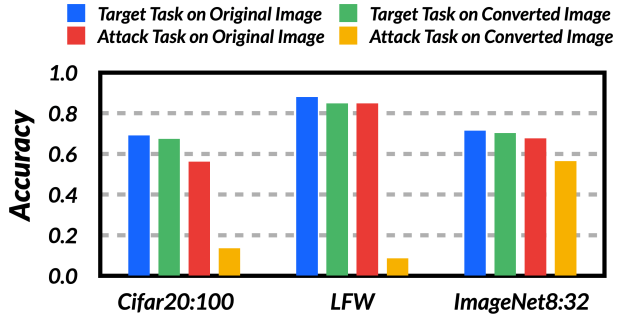

Figure 11: Experiment results of auto recognition attack. These attack tasks have no correlation with targeted task.

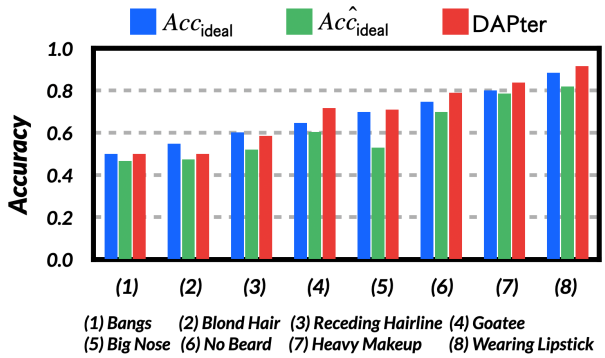

Figure 12: Success rate of auto recognition attack on different attributes (ideal protection v.s. DAPter protection). These attack tasks have correlations with the targeted task.

Case 1. *Attack tasks have no correlation with the targeted task.* To make a comprehensive evaluation, we randomly select two HLDs (Cifar20:100 and ImageNet8:32) and one MLD (LFW) to perform the attack. The attack settings are introduced through the dataset and network architecture pair as follows:

(1) ***Cifar20:100 & ResNet18.*** We use the aggregated superclass label as the targeted task and use the fine label as the attack task.
(2) ***LFW & ResNet18.*** We use gender inference as the targeted task and use the people identification as the attack task.
(3) ***ImageNet8:32 & ResNet50.*** We use the aggregated label used in *ImageNet8* as the target task and use the label of *ImageNet32* as the attack task.

We perform the attack in the following steps: ***(1)*** Transforming the image of each dataset by using the corresponding converters into the converted datasets. ***(2)*** Performing the targeted task and the secondary task on both the original dataset and the converted dataset. The accuracy of the task on each dataset is shown in Figure 11. ***(3)*** Calculating the accuracy loss of the targeted task and the attack task, respectively, before and after the image transformation.

The accuracy of the attack task on the converted image drops ***10***× to ***25***× of that of the targeted task. That is, DAPter can effectively defense auto recognition attacks.

Case 2. *Attack tasks have correlations with the targeted task.* We define a metric *accuracy index* to measure the results of the attack that has correlations with the targeted task. To quantify the correlation between the attack task and the targeted task, we experiment with the attribute inference tasks. For the attribute inference task, which only outputs a result of true or false, the correlation between two inference tasks can be measured with the co-occurrence probability of labels of those two tasks. Let $P_{\text{co}}(A_i, A_j)$ be the co-occurrence probability of attribute A_i, which is the targeted task, and attribute A_j, which is the attack task. The co-occurrence probability can be computed as:

$$Acc_{\text{ideal}} = \left|P_{\text{co}}(A_i, A_j) - 0.5\right| + 0.5 \tag{5}$$

However, in practice, the flaw of dataset and constraint on DL techniques makes it nearly impossible to reach 1.0 accuracy. We measure this impediment by the vanilla accuracy that can be reached by the attacker on the original dataset. So if our DAPter works ideally, the accuracy of the attack task can be computed as:

$$\hat{Acc}_{\text{ideal}} = Acc_{\text{ideal}} \times Acc_{\text{vanilla}} \tag{6}$$

Acc_{vanilla} is the accuracy that can be reached on the original dataset by the same attacker using the same network structure.

We use the dataset CelebA to perform this type of attack, choose a popular gender inference as the targeted task, and train a converter. We use the converter to protect every image in the dataset and train 8 attribute inference tasks with the converted image. The attribute names, corresponding Acc_{ideal} and $\hat{Acc}_{\text{ideal}}$ (see Equation 6), are shown in Figure 12. To ensure that the auto recognition attack has

the same capability as the targeted task, we use the same model structure, i.e., ResNet18, for both the targeted and attack task model. The accuracy of the targeted DLIS (gender inferring) on original images is 0.965 and slightly varies to 0.979 on the converted images. The results of all eight auto recognition attacks are presented in Figure 12. In most cases, the accuracy of the images protected by DAPter is only slightly higher than the theoretical accuracy. Therefore, our DAPter can nearly remove all unrelated information and effectively raise the attack bar, even if the attack task is highly correlated with the targeted task.

7.3.2 Visual Recognition Attack. The targeted DLIS model used here is a gender inference model trained with the CelebA dataset and ResNet18 network. The attack goal is to label another four attributes from the input images of the targeted DLIS model. These attributes are arched eyebrow, bangs, chubby, and wearing glasses. We randomly select twenty positive instances and twenty negative instances of each task from the converted images to organize an evaluation set consisting of 160 images. Then, ten volunteers are invited to conduct the attribute inference task on this dataset. Part of the converted images is shown in Figure 13. The visual recognition attack accuracy on converted images is about 0%. That is, the images protected by DAPter can defend against visual recognition attack and human labeling effectively.

7.3.3 Image Reconstruction Attack. We introduce the image reconstruction attacks in Section 5.3 and perform evaluations on the attack's feasibility by measuring the quality of the reconstructed images in Figure 9(b). Part of the reconstruction attack results has been presented in Figure 9(a). To show the generality of DAPter, here we conduct more experiments on more inference tasks.

We launch this attack on four DLIS models built with the CelebA dataset and ResNet18 network. Some result examples are shown in Figure 13. The average SSIM between original images and reconstructed ones is less than 0.2, indicating that attacks are failed. To further verify these results, we also ask the five volunteers to perform a pairing task. Each person is asked to label 100 reconstructed images with attributes that are not related to the targeted task. In the end, there is no successful case of labeling.

7.4 Usability Evaluation

Backend Throughput. Adapting DAPter does not require any change at the backend side, including the system architecture and inference model. Thus, the service throughput is maintained. However, the TEE-based and FHE-based data protections downgrade the original service throughput by 2.5× ∼ 50× and 1000×, respectively [30]. DAPter shows great superiority in balancing security and usability in our considered scenario.

Service latency. Compared to original service latency, DAPter only adds 109*ms*, 292*ms*, and 309*ms*, respectively, when tested on our three mobile SoCs, i.e., the Snapdragon 855 Plus, Kirin 960, and Helio X30. The user input size in our tests is 224 × 224px.

Bandwidth usage. Thanks to our entropy reduction approach, DAPter can effectively save the network bandwidth usage even compared to the original service case. DAPter's bandwidth efficiency is 2.1× ∼ 41× better than the original one when measured with data from LFW, ImageNet, CelebA, and Cifar10.

8 RELATED WORKS

FHE-based methods protect the input data by performing the DL inference in a homomorphic encryption manner. All user input data is encrypted before uploading to the server side. Work [10] shows that the end-to-end processing time of an FHE-enabled DLIS, equipped with a four-layer neural network, is up to over 297 seconds.

TEE based methods use TrustZone or SGX to protect user input and the whole computation process. Work [14] isolates user's inference data from the DLIS provider. However, SGX is fully operated in CPU and can't utilize the computation power of GPUs, so the inference speed is quite slow. For example, AlexNet takes up to 3,843 seconds to perform one inference. Work [30] combines SGX enclave with GPUs to achieve higher efficiency. These technologies can bring significant system overhead to DLISes. LEAP [29] is a lightweight TEE designed for mobile devices and friendly to DL inference. It makes the DL inference protected by TEE to access mobile GPUs with ease. However, DLIS providers must change their existing DLIS architecture to adopt the above technologies. Recall that our DAPter does not need to change the existing DLIS architecture and only introduce very low system overhead to DLIS (about 109*ms* ∼ 309*ms*).

The DL model partitioning (MP) [19, 23, 32] based solutions split the DL model into two parts and offload the front part, which is responsible for extracting intermediate features from the user input to the client-side. However, the intermediate features appear to be vulnerable to image reconstruction attacks [9]. So MP based methods usually enhance their security with other techniques.

Work [23], an MP-based method, proposes a hybrid architecture for data abuse prevention in the DLIS scenario. Besides network partitioning, this work adopts dimensionality reduction, siamese fine-tuning, and noise addition to cope with image reconstruction attacks. However, only by offloading a large amount of computation, i.e., layers of the neural network, to the user side is it possible to defend against this attack. For example, all front 11 layers of VGG-16 should be offloaded to the user side, and the offloaded layers cost about 7 seconds on the user side, which is larger than 60× that of us.

Work [32] brings differential privacy (DP) into MP-based solutions by adding perturbation to the intermediate feature produced by user-sider layers. However, data abuse prevention cannot be formalized as a DP problem because one simply cannot define adjacent data in this scenario. Therefore, even if someone guarantees it is impossible to infer the specific value of any element in user data through intermediate features, attackers can still mine and restore a large amount of user information through intermediate features. If someone wants to apply DP in our scenario, they must add high-intensity noises to the whole images and inherently cause serious performance degradation to the pre-trained networks. Work [19] also pointed out through experiments that it is difficult for DP-based solutions to balance usability and security in user data protection scenarios.

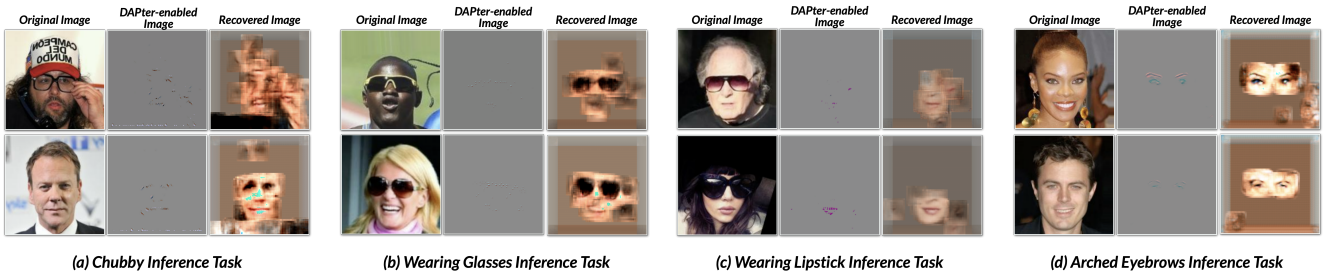

Figure 13: Examples of the image reconstruction attack. For each original image, we show its corresponding two images - one is converted by DAPter and the other one is reconstructed by the adversary from DAPter's converted image.

PAN [19] is a data abuse prevention work that brings adversarial learning to defend against certain input information being abused. It requires users to specify what they want to protect to the user-side model's training procedure. It shows poor scalability in the DLIS scenario because DLIS providers cannot provide a different converter for each user. Please recall that our DAPter can work without the user's involvement/specification, which is more friendly from the web provider's perspective.

PECAM [34] is a privacy protection strategy in video security analytics scenarios. It proposes a securely-reversible transformation for video privacy protection. The protected results produced by PECAM can be inspected by human and AI algorithm to perform common video analytics tasks. Our DAPter is designed with the PoLP principle, which prevents data from being abused in image-based DLIS scenarios. The protected result cannot be restored, nor can it be labeled by human viewers and any secondary AI algorithm tasks.

9 CONCLUSION

This work addresses a unique data abuse issue in the popular scenario of the deep learning inference service. The proposed DAPter is able to intelligently and efficiently convert at the user side the DLIS inputs into their abuse-prevented versions, so that the converted inputs cannot be abused to train new DLIS models at the backend side. DAPter practices the principle of least privilege in the context of DLIS and strikes a balance between security and usability. Our comprehensive evaluation shows that DAPter is lightweight on mobile devices, transparent to existing DLISes, and secure in defending against the adversary. DAPter is able to strengthen the trust between the web providers and their users.

ACKNOWLEDGMENTS

We sincerely thank the reviewers for their valuable comments helping us to improve this work. This work was supported in part by NSFC-61872180, Jiangsu "Shuang-Chuang" Program, and Jiangsu "Six-Talent-Peaks" Program. Fengyuan Xu was supported in part by Ant Financial through the Ant Financial Science Funds for Security Research.

REFERENCES

[1] 2011. Popular Synsets of ImageNet. http://image-net.org/explore.php. [Online; accessed Jan-28-2020].
[2] 2019. Zao's deepfake face-swapping app shows uploading your photos is riskier than ever. https://blogs.lse.ac.uk/medialse/2019/09/10/zaos-deepfake-face-swapping-app-shows-uploading-your-photos-is-riskier-than-ever/. [Online; accessed Jan-28-2020].
[3] 2020. AI's new workforce: the data-labelling industry spreads globally. https://www.ft.com/content/56dde36c-aa40-11e9-984c-fac8325aaa04. [Online; accessed Jan-28-2020].
[4] 2020. Artificial Intelligence as a Service Market by Service Type (Software Tools and Services), Technology (Machine Learning and Deep Learning, and Natural Language Processing), Organization Size, Vertical, and Region - Global Forecast 2023. https://www.marketsandmarkets.com/Market-Reports/artificial-intelligence-ai-as-a-service-market-121842268.html. [Online; accessed Sep-18-2020].
[5] 2020. Face++ Privacy Policy. https://www.faceplusplus.com/privacy-policy/. [Online; accessed Jan-28-2020].
[6] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *USENIX symposium on operating systems design and implementation (OSDI 16)*. 265–283.
[7] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. 2017. Privacy-preserving classification on deep neural network. *IACR Cryptology ePrint Archive* (2017), 35.
[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition*.
[9] Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4829–4837.
[10] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*. 201–210.
[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[12] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2017. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189* (2017).
[13] Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. 2012. Learning to Align from Scratch. In *Neural Information Processing Systems*.
[14] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. 2018. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961* (2018).
[15] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175* (2019).
[16] Jagat Narain Kapur, Prasanna K Sahoo, and Andrew KC Wong. 1985. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing* (1985), 273–285.
[17] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Citeseer.
[18] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* (1998), 2278–2324.
[19] Sicong Liu, Junzhao Du, Anshumali Shrivastava, and Lin Zhong. 2019. Privacy Adversarial Network: Representation Learning for Mobile Data Privacy. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2019).
[20] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer*

Vision.

[21] Fatemehsadat Mirshghallah, Mohammadkazem Taram, Praneeth Vepakomma, Abhishek Singh, Ramesh Raskar, and Hadi Esmaeilzadeh. 2020. Privacy in Deep Learning: A Survey. *arXiv preprint arXiv:2004.12254* (2020).

[22] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1717–1724.

[23] Seyed Ali Osia, Ali Shahin Shamsabadi, Ali Taheri, Kleomenis Katevas, Sina Sajadmanesh, Hamid R Rabiee, Nicholas D Lane, and Hamed Haddadi. 2017. A hybrid deep learning architecture for privacy-preserving mobile analytics. *arXiv preprint arXiv:1703.02952* (2017).

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* (2015), 211–252.

[26] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*. 618–626.

[27] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[28] Jonathan Soifer, Jason Li, Mingqin Li, Jeffrey Zhu, Yingnan Li, Yuxiong He, Elton Zheng, Adi Oltean, Maya Mosyak, Chris Barnes, Thomas Liu, and Junhua Wang. 2019. Deep Learning Inference Service at Microsoft. In *USENIX Conference on Operational Machine Learning*. USENIX Association, 15–17.

[29] Lizhi Sun, Shuocheng Wang, Hao Wu, Yuhang Gong, Fengyuan Xu, Yunxin Liu, Hao Han, and Sheng Zhong. 2021. App Developer Centric Trusted Execution Environment. *arXiv preprint arXiv:2102.02465* (2021).

[30] Florian Tramer and Dan Boneh. 2018. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv preprint arXiv:1806.03287* (2018).

[31] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* (2017).

[32] Ji Wang, Jianguo Zhang, Weidong Bao, Xiaomin Zhu, Bokai Cao, and Philip S Yu. 2018. Not just privacy: Improving performance of private deep learning in mobile cloud. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2407–2416.

[33] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[34] Hao Wu, Xuejin Tian, Minghao Li, Yunxin Liu, Ganesh Ananthanarayanan, Fengyuan Xu, and Sheng Zhong. 2021. PECAM: Privacy-Enhanced Video Streaming and Analytics via Securely-Reversible Transformation. In *The 27th Annual International Conference on Mobile Computing and Networking*.