GAPter: GRAY-BOX DATA PROTECTOR FOR DEEP LEARNING INFERENCE SERVICES AT USER SIDE

Hao Wu, Bo Yang, Xiaopeng Ke, Siyi He, Fengyuan Xu, and Sheng Zhong*

State Key Laboratory for Novel Software Technology, Nanjing University

ABSTRACT

The widespread deployment of Deep Learning Inference Services (DLISes) has raised people's concerns about their data privacy being breached. Although data privacy enhancement has recently attracted a lot of attention, existing solutions all require the cooperation of service providers. Users lose control of their data when making data privacy enhancement decisions. However, it is difficult to enable the user-side control of data abuse prevention because users do not have any programming skills, deep learning knowledge, or rich computing resources. In this work, we propose a fully-automatic user-side data privacy enhancement solution, GAPter, for DLISes. Given such a DLIS, GAPter can adaptively fuzz the service for a suitable enhancement strategy, with no cooperation between the DLIS provider and the user. We have implemented and comprehensively evaluated GAPter. The experimental results show that GAPter can find good balance points between privacy enhancement and user data utility.

Index Terms— Data Privacy, Data Security, Deep Learning Privacy

1. INTRODUCTION

Deep Learning Inference Service (DLIS) [1, 2] lately gains a lot of popularity. It delivers the power of artificial intelligence (AI) to resource-limited devices by offloading expensive computations to the cloud. Leading web service providers, such as Microsoft, Google, Amazon, and Face++, have achieved great business successes with the DLISes launched for their customers.

While enjoying huge benefits of DLIS, people are also deeply concerned about the potential privacy compromise. DLISes, by nature, collect a lot of input data containing much unnecessary information regarding their functions. For example, a DLIS for recognizing vehicle types usually captures visual profiles of nearby pedestrians. These high-resolution profile images, without any authorization, could be freely exploited to infer sensitive information about captured people or train new face recognition models.

The severe data privacy compromise has caught a lot of attention. Laws and regulations have been put in place world-

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

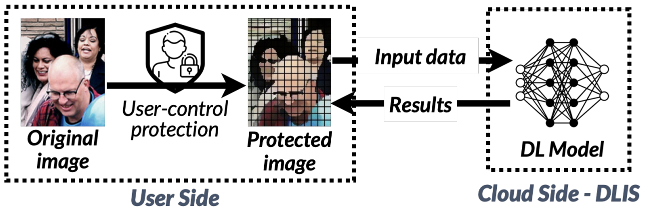

Fig. 1: GAPter can enhance user privacy with almost no impact on DLIS's accuracy.

wide to protect user data fed to the deep learning (DL) models, such as European GDPR [3] and California's CCPA [4]. New technologies are invented to facilitate the installation of these laws and regulations. They [5, 6, 7, 2, 8] effectively alleviate the potential privacy issues in the data collection, processing, and pre-transmission stage respectively. Unfortunately, the success of these solutions requires honest cooperation from DLIS providers, which might not be available in many real-world scenarios like a malicious DLIS provider. Additionally, adopting these solutions requires the deep expertise of DL and a huge amount of computing resources, making large-scale deployment costs quite high.

However, it is challenging to bring the control power of user data back to them, especially when minimizing the prevention impacts on the inference quality of DLIS. *First*, unlike existing solutions, there is no cooperation from DLIS providers. *Second*, the prevention requires auto adaptation for each DLIS without the user's participation. *Last*, we aim to prepare an enhancement solution for a DLIS just on a typical home PC without a powerful GPU.

In this work, we propose GAPter, a user-side privacy enhancement framework against DLIS providers (Figure 1). GAPter protects the DLIS input privacy at the user side and maintains almost the same inference QoS as the original one. We implement the GAPter and experiment with five representative DLISes built from open-source projects. Experiments show that GAPter can achieve a good balance between data privacy enhancement and the DLIS's QoS preservation. Specifically, according to our extensive experiments on facial privacy, it can protect more than 50% of faces at the expense of a 7% drop in data utility.

We highlight our work's contributions as follows:

1. GAPter is the first effort to offer privacy enhancement

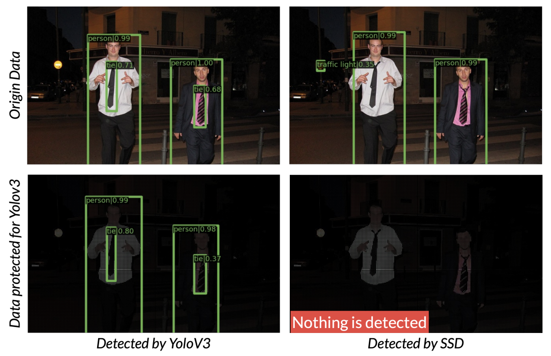

Fig. 2: The protected results for YoloV3, and the inference accuracy of YoloV3 and SSD on the data protected by the same data protector.

at the user side with little QoS compromise. With it, users have the freedom and rights to make prevention decisions.

2. GAPter enables the enhancement via an efficient data protection mechanism for gray-box DLIS models. This mechanism removes the DL task-irrelevant information from the user data through a set of semantic-preserving protection primitives.

3. We implement the GAPter system and show that for popular open-sourced DLISes, GAPter can strike a good balance between the data utility and privacy enhancement.

2. GAPter DESIGN

Design Challenges. There are three challenges for GAPter to produce a data protector in our scenario: *Grey-box Model.* The DLIS can only be used to perform inference. Remote DLIS cannot support back-propagation for user access. *Customized Protection.* As shown in Figure 2, one data protector may have different impacts on different DLISes' inference results. Customizing data protectors for different DLISes is important to maintain the inference accuracy. *Limited Resources.* Data protectors should be generated for users with their limited computation resources, e.g., a CPU-only laptop at home.

GAPter is a user-side technology, which can addressing all the above challenges. The key designs of the generation technology are a set of configurable data protection primitives and a searching algorithm for primitive parameters and combinations. The data protector is a combination of protection primitives with suitable configurations. We demonstrate the workflow of the data protector generation in Figure 3.

2.1. Data Protection Primitives

The heart of the produced data protector is a set of semantic-preserving protection primitives configured with suitable parameters. Each primitive is a basic data protection operation

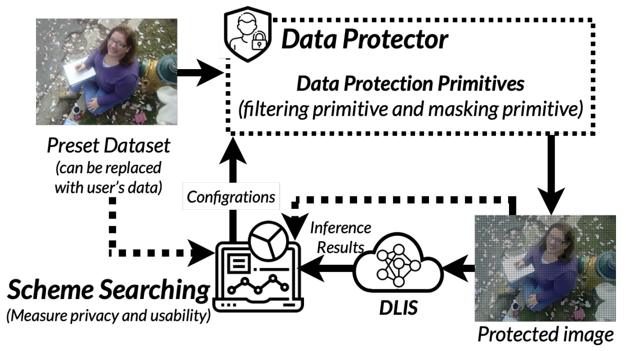
Data Protector
Data Protection Primitives
(filtering primitive and masking primitive)
Preset Dataset
(can be replaced with user's data)
Configrations
Inference Results
Scheme Searching
(Measure privacy and usability)
DLIS
Protected image

Fig. 3: The workflow of GAPter for producing a data protector.

whose protection granularity is configurable through parameters. The protection primitives can maintain the user data's utility and remove as much information from the user data as possible. The data has less information is less likely to be leaked [2]. GAPter provides two primitives, i.e., filtering and masking.

The filtering primitive shares a similar idea of the work PECAM [9]. It is a semantics-preserving image style transformation with an appropriate privacy-enhancement style. This primitive can remove the instance-level details and keep the object's position and category-level information. The protection granularity of the filtering primitive is controlled by the size of the smallest object to protect.

The masking primitive is inspired by the information-dropping data augmentation technique, i.e., grid mask [10]. Among many information-dropping based techniques, the gird mask performs better in semantics-retention by preserving the continuous area as much as possible and is not inclined to delete small objects. The gird mask's protection granularity is controlled by the total masked area. There are two other parameters that affect the protection effect, i.e., the area of a single mask block and the mask's mode[1].

Note that our design is extensible, and more data protection primitives can be added easily. For example, users can add the blurring primitive or the noise-adding primitive to in the future to enhance the data protection effect.

2.2. Protection Scheme Searching

2.2.1. Optimization Objective.

The ideal data protector can balance the protection effectiveness and data's utility. Therefore, we define the usability index (UI) to measure the DLIS's inference accuracy on protected data and the protection index (PI) to measure the protection effectiveness. Finally, we add the two indices together as the optimization objective to search the data protector.

Usability Index (UI) is calculated by dividing the DLIS's inference accuracy on protected data by that on original data

[1]There are two modes of the grid mask. The first mode produces square-like masks. The second mode produces strip-like masks.

(Equation 1).

$$\mathsf{UI} = \frac{mAP_{IoU=0.5}(\mathsf{protect}(\mathsf{data}))}{mAP_{IoU=0.5}(\mathsf{data})} \tag{1}$$

The mAP (mean Average Precision) is the most commonly-used metric to measure the prediction precision in object detection scenario [11]. IoU (Intersection over Union) is the ratio of the area of overlap to the union area between the ground truth and predicted bounding box. $mAP_{IoU=0.5}$ means that we calculate mAP with the predicted bounding boxes whose IoU is larger than 0.5.

Protection Index (PI) is measured by the summation of the weighted granularity of different protection primitives. Recall that a larger protection granularity means more information will be removed from the user data. PI is calculated by $\mathsf{PI} = \lambda \times (\mathsf{f}+\mathsf{m})$. The f and m are the granularity control parameters of the filtering primitive and masking primitive, respectively. The values of these two parameters range from 0 to 1. λ is 0.5 in our context to scale the PI value to between 0 to 1. Then we can generate a suitable data protector for the DLIS by maximizing the value of $\mathsf{UI} + \mathsf{PI}$.

Algorithm 1: Pseudo code to search for optimal data protector parameters.

```
Data: The dataset: imgs; the param searching space:
      $P_{space}$; And the number of max searching steps:
      $S_{max}$.
Result: The configuration of the best data protector: $p_{best}$
1  begin
2      prt = comb(filtering, masking)
3      p = random_sample($P_{space}$)
4      $p_{best}$ = p
5      $Acc_{max}$ = 0
6      for $i \in [0, S_{max})$ do
7          $imgs_p$ = prt(p, imgs)
8          PI = compute_pi(p)
9          UI = compute_ui(imgs, $imgs_p$)
10         Acc = PI + UI
11         p = BO(p, Acc)
12         if $Acc_{max} \leq$ Acc then
13             $Acc_{max}$ = Acc
14             $p_{best}$ = p
15     return $p_{best}$
```

2.2.2. *Searching Algorithm.*

The key step of generating the data protector is configuring the parameters of the filtering and masking primitives. There is only a parameter f in the filtering primitive that controls protection granularity. Besides the parameter m controlling the masking primitive's protection granularity, there are two other parameters s and w to control the area of a single mask block and the way to mask the user data. Once these parameters are determined, GAPter will combine these protection primitives configured with searched parameters as the final data protector.

We present the procedure of searching parameters in Algorithm 1. The algorithm inputs several real-world photos, the parameters' searching space, and the number of the max searching times, and outputs the configuration of the best data protector. GAPter prepares five real-world photos randomly selected from the COCO [12] dataset to search the data protector's configuration. Users can replace the preset data samples with their photos freely. This is because the searching algorithm shares the same idea as metamorphic testing to generate the test oracle.

3. EVALUATION

3.1. Implementation & Settings

Software implementation. We implement the filtering primitive by reproducing the FastPass branch[2] and the DataGen of work PECAM [9]. We implement the masking primitive through the grid mask algorithm implemented by project [13]. The code to generate the data protector is about 800 lines of Python code. The shell script to automate the GAPter pipeline is about 600 LoC.

DLISes to enhance. We evaluate the GAPter with 5 representative DL models (i.e., ssd-mobilenetV1, ssd-mobilenetV2, faster-rcnn-inceptionV2, mask-rcnn-inceptionV2, and efficientdetD2) downloaded from the open-sourced community. The perception modules extracted from these DLISes are denoted as DLIS1$\sim$DLIS5.

Datasets. **COCO**$_{face}$: The subset of COCO val2017 [12] where the images contain faces prepared through a face detection algorithm [14]. It consists of 639 images. **COCO**$^{\#}_{search}$: The subset of COCO$_{face}$ used for data protector generation. The dataset's size is indicated by #. **COCO**$^{\mathsf{P}}_{face}$: The protected version of COCO$_{face}$, where images are protected by the data protector P.

Metrics. We propose two metrics to evaluate the data protectors. The *first* is UI (Equation 1), which can be used to measure how the data protector affects the DLIS's inference accuracy. The *second* is face protection index (FPI), which can measure the protection effect of data protectors by counting how many faces are protected. As presented in Equation 2, the FPI is evaluated by performing face matching on the images before and after protection.

$$\mathsf{FPI} = 1 - \frac{\sum(\mathsf{face\_match}(\mathsf{data}, \mathsf{protect}(\mathsf{data})))}{\sum(\mathsf{face\_detect}(\mathsf{data}))} \tag{2}$$

We utilize the commonly used face matching algorithm [14] as the face matching and face detection methods in the following experiments.

[2]Note that the results transformed by FastPass are not reversible.

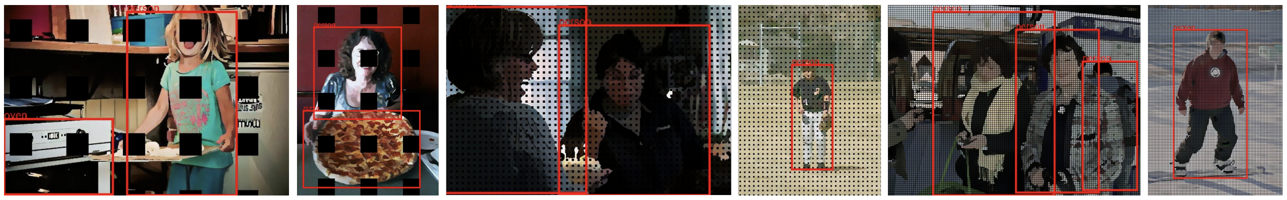
Fig. 4: Samples protected by GAPter for different DLISes.

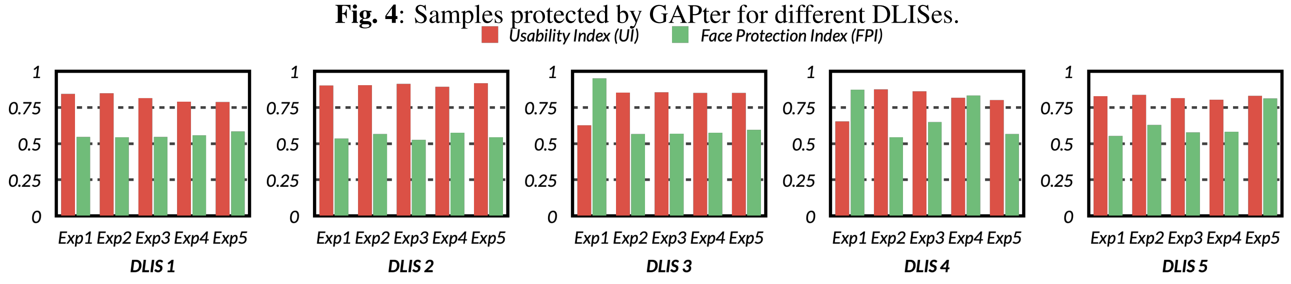

Fig. 5: The effectiveness of the generated data protectors in terms of UI and FPI.

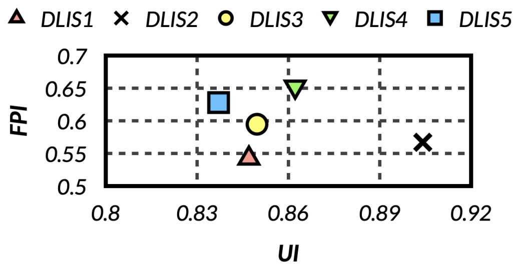

Fig. 6: Sweet points for different DLISes.

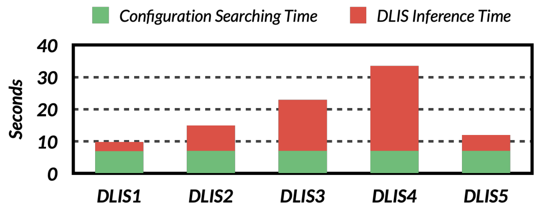

Fig. 7: Time used for each round of data protector generation.

3.2. Experimental Results

Overall Protection Effectiveness. We have made a comprehensive analysis of protection effectiveness in terms of utility and effectiveness. First, we generate data protectors for DLIS1∼5 with Algorithm 1. The S_{max} is set to 200, and the imgs are the dataset COCO$^{5}_{search}$. We generate data protectors for each of the DLISes with its five searched configurations. We evaluate the data protector of each DLIS on the COCO$^{\mathrm{P}}_{face}$ dataset. and evaluate these protectors in terms of UI and FPI.

We present the results in Figure 5. It shows that GAPter can produce data protectors that perform well in balancing data protection and utility retaining for DLIS1∼5. We also report sweet points of the data protector in terms of utility and effectiveness in Figure 6. In our context, the sweet point represent a data protector for the DLIS that maximizes the value of UI + PI. Although GAPter produces protectors for different DLISes with different sweet points, for different DLISes, there is a data protector that can protect more than 50% of the faces at the cost of less than 7% loss of accuracy.

Protected Samples. In Figure 4, we show some samples protected by different protectors generated for different DLISes. We also visualize the DLIS's inference results, i.e., bounding boxes and categories, on the protected data. Recall that the GAPter prevents data from being abused by removing DLIS-related information, not just protecting faces. Welcome to our **anonymous website:** `https://sites.google.com/view/for-icassp2023-review` for more samples.

System Performance. We measure the performance of data protector generation with DLIS1∼5 and COCO$^{5}_{search}$. The data protector's preparation time consists of two parts, i.e., configuration searching time and DLIS inference time. We report the time used for each round in Figure 7. It takes about 7 seconds for GAPter to search for configuration and update the data protector. The rest time is for DLIS inference on the laptop. In our experiments, the number of the searching rounds is 200. So the configuration searching takes about 24 minutes during the data protector generation.

The data protector only takes up 5MB extra storage. It takes about 50ms to protect the user data on mobile device, which is a Google Pixel5 in our experiment. The data privacy enhancement time only takes about 7.5% that of the original DLIS inference on average.

4. CONCLUSION

In this work, we propose a user-side automatic framework, GAPter, to enable the user-side data privacy enhancement and maintain the DLIS's QoS. We have implemented and experimentally show that the GAPter can find good balance points between data privacy enhancement and inference accuracy. GAPter is also a data-driven grey-box automation and paves a new way to perform non-intrusive DL model fuzzing.

5. REFERENCES

[1] 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, "Deep learning inference service at microsoft," in *USENIX Conference on Operational Machine Learning*. 2019, pp. 15–17, USENIX Association.

[2] Hao Wu, Xuejin Tian, Yuhang Gong, Xing Su, Minghao Li, and Fengyuan Xu, "Dapter: Preventing user data abuse in deep learning inference services," in *Proceedings of the Web Conference*, 2021.

[3] Proton Technologies AG, "Complete guide to gdpr compliance," `https://gdpr.eu/`, 2018.

[4] State of California Department of Justice, "California consumer privacy act (ccpa)," `https://oag.ca.gov/privacy/ccpa`, 2018.

[5] Franziska Roesner, David Molnar, Alexander Moshchuk, Tadayoshi Kohno, and Helen J Wang, "World-driven access control for continuous sensing," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[6] Jinhan Hu, Andrei Iosifescu, and Robert LiKamWa, "Lenscap: split-process framework for fine-grained visual privacy control for augmented reality apps," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021.

[7] Suman Jana, Arvind Narayanan, and Vitaly Shmatikov, "A scanner darkly: Protecting user privacy from perceptual applications," in *2013 IEEE symposium on security and privacy*, 2013.

[8] Sicong Liu, Junzhao Du, Anshumali Shrivastava, and Lin Zhong, "Privacy adversarial network: representation learning for mobile data privacy," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2019.

[9] Hao Wu, Xuejin Tian, Minghao Li, Yunxin Liu, Ganesh Ananthanarayanan, Fengyuan Xu, and Sheng Zhong, "Pecam: Privacy-enhanced video streaming and analytics via securely-reversible transformation," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2021.

[10] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia, "Gridmask data augmentation," 2020.

[11] Rafael Padilla, Sergio L Netto, and Eduardo AB da Silva, "A survey on performance metrics for object-detection algorithms," in *International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*, 2014.

[13] dvlab research, "Gridmask data augmentation," `https://github.com/dvlab-research/GridMask`, 2019.

[14] ageitgey, "Face recognition," `https://github.com/ageitgey/face_recognition`, 2017.