# Attribute Inference Privacy Protection for Pre-trained Models

by

Hossein Abedi Khorasgani

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
August 2023

Thesis advisor

**Dr. Yang Wang, Dr. Noman Mohammed**

Author

**Hossein Abedi Khorasgani**

**Attribute Inference Privacy Protection for Pre-trained Models**

# Abstract

With the increasing popularity of machine learning in image processing, privacy concerns have emerged as a significant issue in deploying and using ML services. However, current privacy protection approaches often require computationally expensive training from scratch or extensive fine-tuning of models, posing significant barriers to the development of privacy-conscious models, particularly for smaller organizations seeking to comply with data privacy laws. In this thesis, we address the privacy challenges in computer vision by investigating the effectiveness of two recent fine-tuning methods, Model Reprogramming and Low-Rank Adaptation. We adapt these techniques to provide attribute privacy protection for pre-trained models, minimizing computational overhead and training time. We integrate these methods into an adversarial min-max framework, allowing us to conceal sensitive information from feature outputs without extensive modifications to the pre-trained model, but rather focusing on a small set of new parameters. We demonstrate the effectiveness of our methods by conducting experiments on the CelebA dataset, achieving state-of-the-art performance while significantly reducing computational complexity and cost. Our research provides a valuable contribution to the field of computer vision and privacy, offer-

ing practical solutions to enhance the privacy of machine learning services without compromising efficiency and security.

# Contents

# List of Figures

# List of Tables

# Acknowledgments

*This thesis is dedicated to nothing in particular.*

# Publications

Some content of this thesis, including figures and tables have appeared in the following publication:

- **H. Abedi Khorasgani**, Y. Wang and N. Mohammed. Black-Box Attribute Inference Protection With Adversarial Reprogramming. In International Conference on Privacy, Security and Trust (PST), 2023.

# Chapter 1

# Introduction

Deep learning has been greatly successful in a variety of computer vision applications, such as attribute classification [Liu et al., 2017], semantic segmentation [Yu et al., 2018], and face recognition [Schroff et al., 2015]. However, as the use of ML models becomes more common for image processing, privacy concerns in ML are drawing more attention.

Previous works have shown that ML models can leak unintended information about their dataset or input [Song and Shmatikov, 2019; Melis et al., 2019]. Although a model is trained for a specific task, the model's outputs (e.g., feature embedding) can contain sensitive information about the data sample that can potentially be accessed by an adversary.

For example, consider a scenario where a service provider utilizes a feature extractor on client devices to embed facial expressions such as smiling or frowning in a picture and sends the extracted features to a cloud provider for further processing and improving its services. Even though the model has been trained for a specific task,

its extracted features can still contain sensitive information about the original image, such as the person's gender, age or race. As a result, sharing or storing the extracted features poses a privacy risk as it can be leaked due to security vulnerabilities or be abused by the provider itself [Koczkodaj et al., 2019; Meredith, 2018; Williams, 2019]. This unintended information leakage is a privacy risk that enables adversaries to perform various attacks and learn private information about the users (See Fig. 1.1). Data protection laws and regulations, such as GDPR, further complicate these issues and incite designers to provide models with privacy protection.

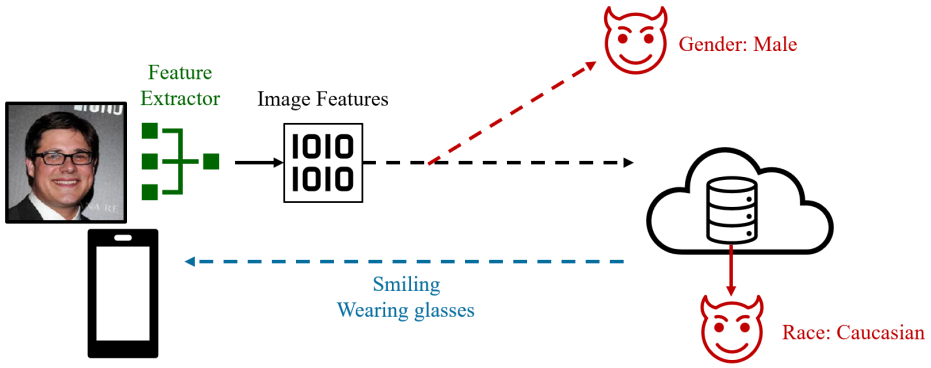



Figure 1.1: Privacy Leak From Extracted Features

This figure illustrates the potential privacy leak from extracted features. The service provider sends the feature embedding to a cloud location for further processing and/or storage. If the attackers gain access to the unprotected features (e.g. through intercepting the data or a vulnerability in the cloud), they are able to infer the private attributes of the image. The figure uses a sample image from CelebA dataset [Liu et al., 2015].

One of the popular ways to address this issue is to train the model with an adversary to provide protection [Ding et al., 2022; Li et al., 2021; Xiao et al., 2020; Wu et al., 2022; Li et al., 2020]. That is, during the initial training of the model, the model is trained in a min-max framework against an adversary trying to infer sensitive attributes. An Alternative approach focuses on minimizing information

extracted (e.g., minimizing mutual information) so that there is no extra information available to be exploited [Mireshghallah et al., 2021; Dave et al., 2022]. While these studies have achieved notable results in performance and privacy, nearly all of them share the implicit assumption that the model is trained from scratch, which can be extremely expensive in terms of computation and time. Furthermore, this assumption makes it harder for smaller organizations to compete with larger firms as they have fewer resources available.

To overcome the barrier of high training costs, many transfer learning models have been developed to enable fine-tuning or repurposing pre-trained models for various downstream tasks with significantly fewer resources [Hinton et al., 2015; Zhou et al., 2017; Guo et al., 2019; Zhang et al., 2019; Tsai et al., 2020]. However, these studies mainly focus on achieving better performance and not privacy issues in ML. A pre-trained model may not have been trained to provide any form of attribute inference protection. This lack of protection can be either from the change in task domain (e.g., gender is not considered a sensitive attribute in the original task, but it is for the downstream task), or simply from the lack of concern of the third party to provide attribute privacy protection. Regardless of the reason, this results in a vulnerable model, enabling adversaries to infer sensitive information from its extracted features. Despite these needs, there are very few works that attempt to address all these constraints (adding privacy protection, using a pre-trained model and having low costs) simultaneously.

To address this gap, we propose investigating the use of novel fine-tuning methods, namely Model Reprogramming and Low-Rank Adaptation, for providing privacy on

pre-trained models with low costs (See Fig. 1.2). More specifically, we propose two schemas: one for utilizing Model Reprogramming in a black-box setting and another for employing Low-Rank Adaptation in a white-box setting.

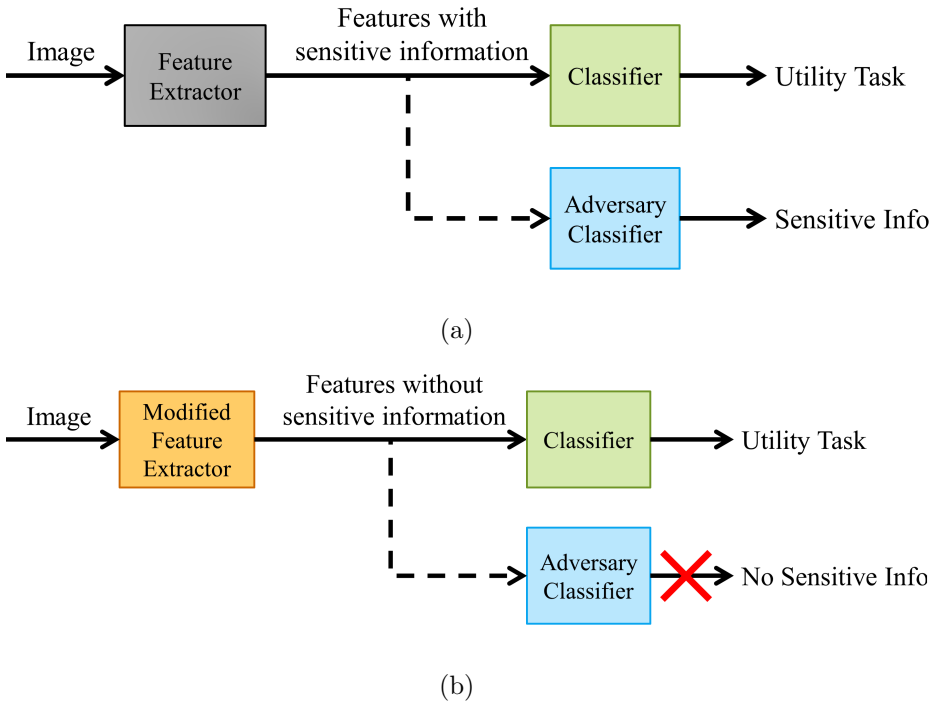



Figure 1.2: Adversarial Attack on Extracted Features

The top figure shows an example scenario where extracted features leak sensitive information because the pre-trained model does not provide attribute inference protection. The bottom figure shows the desired solution, modifying the feature extractor in a manner to remove sensitive information from the extracted features.

Model Reprogramming, also known as Adversarial Reprogramming [Elsayed et al., 2018], is an emerging branch in ML where a model is repurposed for a new task without the need to retrain or fine-tune the original model. Instead, this approach modifies the inputs of the model to utilize it for a new adversarial task. Since compared to retraining or fine-tuning, Model Reprogramming has a lower computational cost and requires less access to the model parameters, it has already been extended for domain

adaptation [Chen, 2022], knowledge transfer [Tsai et al., 2020] and eliminating bias in models [Zhang et al., 2022].

Low-Rank Adaptation (LoRA) is also a recent fine-tuning approach, popular with Large Language Models (LLM) as it provides an extremely efficient method to fine-tune large models with only training a small number of parameters which can be stored separately for different downstream tasks. Instead of fine-tuning a large matrix, LoRA adds two smaller matrices parallel to the original whose output has the same dimension. LoRA then fine-tunes the model by training the new side matrices rather than the original matrix [Hu et al., 2021]. The low overhead approach of LoRA has made it a popular choice for performing fine-tuning, transfer learning and even providing privacy on Large Language Models [Yu et al., 2021; Liu et al., 2022a; Guo et al., 2022; Wang et al., 2023; Treviso et al., 2023; Zhu et al., 2023]. While LoRA has shown great application in LLMs, its presence in computer vision and Convolutional Neural network (CNNs) is severely limited.

In light of the proven utility of both fine-tuning methods in transfer learning, a critical question arises: Can these methods be effectively repurposed to address the challenge of privacy in pre-trained models? We aim to explore and demonstrate the feasibility of utilizing Model Reprogramming and Low-Rank Adaptation techniques for providing privacy protection. In this study, we seek to fill the gap between efficient transfer learning and robust privacy protection, thereby contributing to the advancement of privacy-conscious machine learning services.

## 1.1 Thesis Objective

This thesis aims to investigate the effectiveness and practicality of two recent fine-tuning techniques, Model Reprogramming and Low-Rank Adaptation, in providing attribute inference protection for pre-trained models. Specifically, we examine the utility of Model Reprogramming in a black-box setting and Low-Rank Adaptation in a white-box setting to offer robust protection.

### 1.1.1 Black-Box Protection

In the black-box setting, we assess the effectiveness of Model Reprogramming – which operates at the input level – for providing privacy protection on pre-trained black-box models. Further details of these experiments are available in Chapter 3.

### 1.1.2 White-Box Protection

To address white-box scenarios, we introduce an extension of Low-Rank Adaptation specifically designed for Convolutional Neural networks. This extension facilitates a fine-tuning schema comparable to the methods commonly employed by Large Language Models. Subsequently, we explore the applicability of the LoRA approach for implementing privacy-preserving fine-tuning on pre-trained vision models. Details of this study are presented in Chapter 4.

## 1.2 Contributions

We demonstrate our proposed methods can achieve better or comparable privacy protection trade-offs compared to other state-of-the-art methods with a significantly lower training and computational overhead. To the best of our knowledge, our work is the first to investigate the use of Model Reprogramming and LoRA for providing attribute inference protection in computer vision.

We summarize our contributions as follows:

- We propose a min-max framework with Model Reprogramming for providing attribute inference protection for black-box models.
  - We demonstrate the effectiveness of Model Reprogramming on the CelebA dataset compared to other state-of-the-art methods.
  - We show that our method trains nine times faster than state-of-the-art methods while providing similar privacy protection.
- We propose the use of LoRA for fine-tuning pre-trained CNNs in a white-box setting for providing attribute inference protection.
  - We evaluate the effectiveness of LoRA on the CelebA dataset compared to other state-of-the-art methods.
  - We show that LoRA can provide superior utility and privacy protection with training less than 1% number of parameters compared to the original model.

## 1.3 Thesis Structure

This thesis is organized as follows: In Chapter 2, we give an overview of relevant background information on reprogramming, LoRA and adversarial training, followed by literature review in computer vision privacy. We provide our black-box approach with Model Reprogramming in Chapter 3 and our white-box approach with LoRA in Chapter 4. Finally, we conclude the thesis and propose future works in Chapter 5.

# Chapter 2

# Background and Related Works

## 2.1 Background

### 2.1.1 Model Reprogramming

Model Reprogramming, also known as Adversarial Reprogramming, is a form of adversarial attack on ML models that transforms the inputs for model to change the model's behavior to a desired adversarial behavior [Elsayed et al., 2018]. Transforming the inputs for images, also known as visual prompting, is usually achieved by adding values to whole or parts of the image or padding the image with custom values [Elsayed et al., 2018; Zheng et al., 2021; Jia et al., 2022; Kim et al., 2022; Bahng et al., 2022; Zhang et al., 2022].

Since Model Reprogramming only relies on learning a simple transformer for input data and does not modify the actual model, it has quickly gained interest as an alternative for transfer learning, fine-tuning and domain adaptation [Tsai et al., 2020;

Hambardzumyan et al., 2021; Bahng et al., 2022; Chen, 2022; Zhang et al., 2022].

In a recent work, Zhang et al. further investigates the robustness of Model Reprogramming by using it to boost a pre-trained model's fairness [Zhang et al., 2022]. Their method uses Model Reprogramming in an adversarial framework to improve the fairness of the pre-trained model without retraining or fine-tuning the pre-trained model.

### 2.1.2 Low-Rank Adaptation

Low-Rank Adaptation for Large Language Models was initially proposed by Hu et al. [2021], as fine-tuning method with significantly lower computational overhead while keeping the model's quality intact. Following this, LoRA has been widely adapted for transfer learning, fine-tuning and domain adaptation for LLMs [Wang et al., 2023; Treviso et al., 2023; Pfeiffer et al., 2023; Liu et al., 2022a; Guo et al., 2022; Zhu et al., 2023].

Despite the proven success of LoRA in Large Language Models, its current popularity for use in Convolutional Neural networks remains limited. Nonetheless, it is worth noting that a related concept to LoRA has been previously explored, although not for fine-tuning purposes, but rather for approximating and accelerating CNNs [Jaderberg et al., 2014; Chang and Sha, 2017]. Building upon the achievements of LoRA in LLMs and the existence of a parallel idea in CNNs, a natural question arises: Can LoRA fine-tuning be effectively applied in CNNs, and how might it contribute to privacy preservation in pre-trained models?

### 2.1.3 Adversarial Training / Min-Max Framework

In this section, we define the problem statement by formulating the utility and privacy problems and defining the min-max framework used for training our models. Let $\mathcal{D}$ be our training dataset with color images (with size $N \times N$) as input features $X \in \mathbb{R}^{N \times N \times 3}$, $M$ task labels $Y \in \mathbb{R}^M$, and $P$ sensitive (private) labels $S \in \mathbb{R}^P$.

We denote the feature extractor as $f_F : X \to Z$ which outputs features embedding $Z \in \mathbb{R}^E$, feature classifier $f_C : Z \to Y$ which outputs classifier scores $Y \in \mathbb{R}^M$ based on extracted features, and the adversary classifier is $f_A : Z \to S$ which attempts to infer private attributes from the feature embedding.

Our goal is to train $f_F$ and $f_C$ in a way that the adversary is unable to extract sensitive information from the extracted features (output of $f_F$) while the classifier is still able to perform the utility task correctly (see Fig. 1.2(b)).

**Utility Task:** The utility task which involves both the feature extractor and classifier is finding the correct label scoring $y = f_C(f_F(x; \theta_F); \theta_C)$ where $x \in X$ and $\theta_F$ and $\theta_C$ are the model parameters for the feature extractor and classifier respectively. We define the utility task as a multi-class multi-label classification problem which we measure with mean of Binary Cross Entropy loss over all labels, denoted as $\mathcal{L}_{Task}$.

**Adversarial Classifier:** An attacker can train an adversarial model for detecting private attributes by using unprotected feature embedding $s = f_A(f_F(x; \theta_F); \theta_A)$ where $x \in X$ and $\theta_A$ is the model parameters of the adversary. Similar to the utility task, the loss of the adversarial classifier is measured with the mean of Binary Cross Entropy loss over all labels $S$, denoted as $\mathcal{L}_{Adv}$.

**Min-Max Optimization:** To train our model with two competing objectives, we use a min-max optimization framework and define the loss as:

$$\mathcal{L}_{MinMax} = (1 - \lambda)\mathcal{L}_{Task} - \lambda \cdot \mathcal{L}_{Adv} \tag{2.1}$$

Where $\lambda \in (0, 1)$ is a privacy-utility trade-off knob specifying which training objective, utility or privacy protection, is more important during training. Higher values prioritize privacy protection, and lower values focus more on utility. Note that the negative sign in (2.1) before $\mathcal{L}_{Adv}$ signifies maximizing $\mathcal{L}_{Adv}$ while minimizing $\mathcal{L}_{Task}$.

Our Min-Max training algorithm is shown in Algorithm 1. As noted, the training contains three objectives: the classifier minimizing $\mathcal{L}_{Task}$, the adversary minimizing $\mathcal{L}_{Adv}$ and the feature extractor minimizing $\mathcal{L}_{Task}$ and maximizing $\mathcal{L}_{Adv}$ simultaneously (i.e., minimizing $\mathcal{L}_{MinMax}$). Our algorithm also includes an additional parameter $k$ denoting the number of steps the adversary is trained before we train the feature extractor and classifier. The reason for that is that by training the adversary more, we achieve a more robust model which can better help us with providing protection.

Both our studies utilize this Min-Max framework and algorithm for training our proposed method. Any needed modification is noted in the relevant chapter.

## 2.2 Related Works

As noted, various defense methods have been developed focusing on different aspects of privacy. In this section, we present relevant literature on various privacy protection mechanisms and note their capabilities and limitations.

**Algorithm 1** Min-Max Training Algorithm

**Input:** $k$, $f_F$, Dataset $\mathcal{D}$

**Output:** $\theta_F$, $\theta_C$

**for** *every epoch* **do**
  **while** *training batches exist* **do**
    **for** $k$ steps **do** ▷ Train Adversary for $k$ batches
      **sample new batch** from $D$
      $\mathcal{L}_{Adv} \rightarrow$ update $\theta_A$
    **end for**
    **sample new batch** from $D$
    $\mathcal{L}_{MinMax} \rightarrow$ update $\theta_F$
    **sample new batch** from $D$
    $\mathcal{L}_{Task} \rightarrow$ update $\theta_C$
  **end while**
**end for**

### 2.2.1 Differential Privacy

Differential Privacy (DP) [Dwork and Roth, 2014] is currently one of the most recognized methods for providing privacy. DP is based on a mathematical notion of privacy wherein the difference between two models trained on datasets $D$ and $D'$, where $D$ and $D'$ only differ in one record, is limited by a privacy parameter $\epsilon$. Differentially-Private Stochastic Gradient Descent (DP-SGD) [Abadi et al., 2016] is one of the common implementations of DP which operates by adding random Gaussian noise to the gradients during training. While DP is widely used for many

applications and can also defend against various attacks [Liu et al., 2022b; Hu et al., 2022], its privacy guarantee comes with a heavy utility trade-off cost.

### 2.2.2 Privacy Preserving Feature Extraction

P-FEAT, proposed by Ding et al. [2022] is one of the more recent studies on training a feature extractor against an adversary to remove sensitive or extra information from the extracted features [Ding et al., 2022; Li et al., 2021; Xiao et al., 2020; Li et al., 2020]. P-FEAT propose a method to train a feature extractor based on adversarial training to ensure the extracted features do not contain sensitive attributes and cannot be used by an adversary for inference attacks. Although these studies provide good utility and privacy trade-offs, they all suffer from the major drawback, requiring white-box access as the model is being trained from scratch, making these approaches prohibitively expensive in terms of computational power and time.

### 2.2.3 Image Obfuscation

In contrast to training a private feature extractor, image obfuscation methods use a complex image degradation function for transforming the input data [Dave et al., 2022; Wu et al., 2022, 2018]. Such methods focus on training an anonymizer and feature extractor pair for transmitting anonymized data from clients to servers. While these methods do not focus on a black-box setting, their approach can be adapted for a black-box scenario. Wu et al. propose a supervised adversarial training framework to remove features from images [Wu et al., 2022]. In their approach, an image degradation model is trained against an adversary which attempts to infer sensitive

attributes from the degraded image. Dave et al. proposed Self-supervised Privacy Preservation for Action Recognition (SPAct), a self-supervised version of [Wu et al., 2022], where the input is degraded to remove any unnecessary information that can leak privacy. They train the degradation model by using Contrastive Self-supervised Loss. [Dave et al., 2022] Effectively, Contrastive Self-supervised Loss ensures that two similar images result in a similar output, while different images output vastly different results.

Cloak proposed by Mireshghallah et al. [2021], provides a self-supervised feature suppression framework that operates by determining the features that most contribute to the model's utility and suppressing all other features. Cloak separates relevant and irrelevant features by how much they degrade utility with added noise. Features that can tolerate higher values of noise are classified as non-conducive features. After finding non-conducive features, Cloak replaces them with a constant value trained to maximize utility. Unlike other methods, Cloak does not provide privacy using a simulated adversary. Instead, by removing features that do not contribute to the final task, Cloak removes information that can be exploited in an attack. Compared to other methods, Cloak has less computational complexity as it does not use any complex architecture like CNNs. In addition, to the best of our knowledge, Cloak is the only work that attempts to address the issue of privacy in black-box models. However, Cloak anonymizes the image by removing information as much as possible, and does not directly protect sensitive attributes, which can result in sub-optimal protection.

## 2.3 Summary

In this chapter, we introduced two recent methods popular for fine-tuning models,Model Reprogramming and Low-Rank Adaptation, followed by a generic Min-Max framework that is commonly used for training a model against two competing objectives. Furthermore we reviewed the recent literature in privacy for computer vision and highlighted their shortcoming, notably their high costs for training a private model.

# Chapter 3

# Black-Box Protection With Adversarial Reprogramming

With the increased popularity of ML models, privacy concerns in ML are getting more attention, creating a need for providing better protection during the utilization of machine learning services. Despite the advancements in machine learning techniques, prevailing privacy methods often necessitate extensive retraining, from scratch or through fine-tuning large portions of the model. This shortcoming acts as a bottleneck in the progression of privacy in models, impeding their development due to considerable computational and time requirements. To address this gap, we study the potential of Model Reprogramming as a means of providing privacy on a pre-trained model in a black-box setting. Providing privacy in a black-box setting allows us to utilize robust commercial ML services without concern over whether they provide privacy as we implement our own privacy protection via Model Reprogramming. More specifically, we adapt Model Reprogramming into an adversarial min-max framework

aiming to obfuscate sensitive information in feature outputs without modifying the original model and with significantly lower computational overhead.

## 3.1 Method

### 3.1.1 Reprogramming Model

The reprogramming model is a simple model that combines an input image with a set of custom values (the adversarial program), creating a modified input which is then forwarded to the target model. A combination of the input and the adversarial program can be done in a variety of manners, such as replacing a part of the image, padding the image with custom values or adding the image and custom values together. The adversarial program is optimized (usually through backpropagation) to realize the desired behavior from the pre-trained model.

The adversarial program, also known as visual prompt or reprogram values, is a global adversarial perturbation. In other words, the values are not dependent on the input image, rather the same program is used for any input image. Given that use of a global perturbation may not seem intuitive (as features in an image can have varied locations), we will provide a more detailed explanation of how reprogramming works in Section 3.1.2. Additionally, reprogramming can include a label mapping from the original task to the adversarial task. However, as we only investigate the use of reprogramming in the same domain, we do not use any label mapping.

In this chapter, we define our reprogramming model as the sum of the input and the adversarial program without any modifications on the original image. Following

previous works, we use *tanh* to restrict the reprogram values to $(-1, 1)$, same range as the normalized inputs for the model (See Fig. 3.1). More formally we define our reprogramming model as:

$$f_R(x; \theta_R) = x + tanh(\delta) \tag{3.1}$$

where $\delta$ is the adversarial program ($\delta \in \mathbb{R}^{N \times N \times 3}$), which is part of the reprogramming model's parameters ($\theta_R$). Note that since we do not apply any transformation on the image (such as padding or resizing), our model does not have any other hyperparameters, and the adversarial program is the only trainable part of our model.

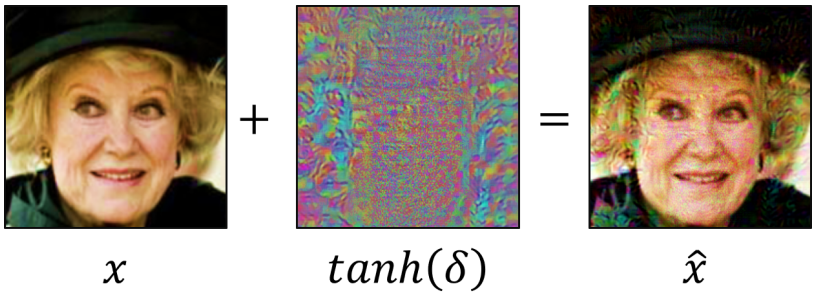



Figure 3.1: Sample Adversarial Program

Our adversarial program adds a set of values to the input image. In our method, we use a program with the same size as the input image, without any masking or padding operation. We use *tanh* to limit the values added on the image to $(-1, 1)$. This figure shows the transformer process applied to a sample image from CelebA [Liu et al., 2015].

### 3.1.2 How Reprogramming Works

Working of adversarial reprogramming is currently an active topic of investigation. In the original paper, Elsayed et al., justify the working of an adversarial program as a result of the nonlinearity of the model [Elsayed et al., 2018]. That is, assuming a linear model with weights $\theta$ and bias $\beta$, the modified input $x + \delta$ will change the

output of the model from $x\theta + \beta$ to $x\theta + \delta\theta + \beta$. The $\delta\theta$ effectively acts as a constant and can manipulate the bias in the model, however, the adversarial program is not able to modify the calculation done on the input ($x\theta$). Elsayed et al. then conclude that a requirement for adversarial reprogramming to be successful is for the model to contain nonlinear calculations, which is satisfied by the nonlinear deep network.

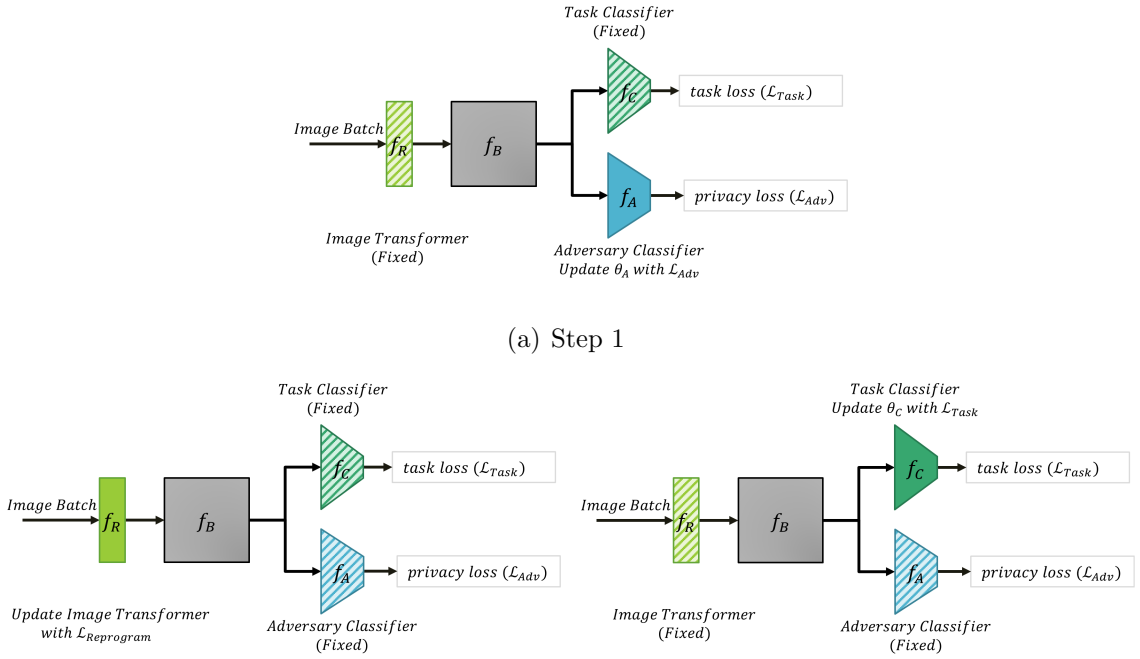
Zheng et al. carried out a more detailed investigation into the operation of adversarial reprogramming by analyzing the similarity between source and target domains [Zheng et al., 2021]. Specifically, Zheng et al. analyze the gradient since and alignment of inputs during training and their interaction on the ML model.

Since our approach does not include source or domain shift, we believe the nonlinearity explanation of [Elsayed et al., 2018] is more appropriate in this setting. It should be noted that we are not using reprogramming to obfuscate or anonymize the image. We use reprogramming to manipulate the operation of the pre-trained model. More specifically, we remove sensitive information from the extracted features by exploiting the model's nonlinearity, effectively suppressing calculations related to private attributes throughout the model.

### 3.1.3 Min-Max Framework

To adapt the Min-Max framework presented Section 2.1.3, we denote the Min-Max loss as $\mathcal{L}_{Reprogram}$ and $f_F(\cdot;\theta_F)$ as $f_R(f_B(\cdot);\theta_R)$ where $f_B$ is the pre-trained black-box model. An overview of our algorithm is shown in Fig. 3.2. In step 1, shown in Fig. 3.2(a), we fix the reprogram values and the classifier and update the adversarial classifier ($\theta_A$) to minimize $\mathcal{L}_{Adv}$ for $k$ batches. Next step, shown in Fig. 3.2(b), we

update the input transformer ($\theta_R$) to minimize $\mathcal{L}_{Reprogram}$, while keeping classifier and adversarial classifier fixed. Finally, in step 3, Fig. 3.2(c), we update the classifier ($\theta_C$) to minimize $\mathcal{L}_{Task}$. We repeat this process for the whole epoch. Here, we use uniform distribution ranging from -1 to 1 to initialize $\delta$.




(a) Step 1

(b) Step 2

(c) Step 3

Figure 3.2: Three-Step Reprogramming Algorithm

Overview of our training algorithm. $f_B$ is the black-box feature extractor, $f_C$ is the feature classifier, $f_A$ is the adversarial classifier, and $f_R$ is our reprogramming model. The training framework is a min-max optimization process with 3 distinct training steps. In step 1, the adversarial classifier is optimized for inference for $k$ batches ($\mathcal{L}_{Adv}$), while the other models remain fixed (frozen). In step 2, the classifier and adversarial classifier are fixed, and the reprogramming model is optimized with the new $\mathcal{L}_{Reprogram}$ loss for 1 batch. Finally, in step 3, the classifier is optimized for the task utility ($\mathcal{L}_{Task}$). For more details, refer to Section 3.1.

## 3.2 Experiments

In this section, we evaluate our proposed method's performance on the popular CelebA dataset [Liu et al., 2015] and compare it with existing solutions in the literature. Given that methods that can be applied to a black-box scenario are limited, we also compare our method with other white-box methods that offer privacy protection.

**Datasets:** For the dataset, we use the popular CelebA [Liu et al., 2015] dataset – more specifically, the aligned and cropped faces – containing 160K training and 20k test images with 40 binary facial attributes. We define the utility task as detecting arched eyebrows, black hair, high cheekbones and smile. We also define eyeglasses, heavy makeup, gender and age as sensitive attributes. The dataset images are resized to $224 \times 224$ and normalized for training. For our purposes, we split the training set into two disjoint sets $\mathcal{D}_p$ and $\mathcal{D}_f$ with 100k and 60k images respectively. We use $\mathcal{D}_p$ for pre-training the feature extractor (i.e., the black-box model) and $\mathcal{D}_f$ for providing privacy protection without altering the black-box model. Finally, we use the test set for evaluation.

We chose the CelebA dataset because it has been utilized in previous works covering attribute inference and privacy on images. Using a dataset that has been extensively studied ensures comparability and enables meaningful comparisons with existing studies.

**Comparison:** We compare the performance of our method with state-of-the-art methods, a few of which can be applied in a black-box setting and the rest we compare as a white-box baseline. The black-box comparisons are: **Cloak** [Mireshghallah et al.,

2021], **SPAct** [Dave et al., 2022], **Obfuscator** (Supervised adversarial framework) [Wu et al., 2022] and **Downsampling** the input image by factor of 2 and 4. We also compare with DP-SGD (referred to as **DP**) [Abadi et al., 2016], as a white-box baseline.

To adapt Obfuscator and SPAct for our black-box setting, we split the utility model with our black-box feature extractor and a classifier and train the classifier instead of the feature extractor when necessary. Overall, this results in a similar framework to Fig. 3.2, where the reprogramming model is replaced by the image degradation model used in SPAct or Obfuscator. Note that for DP-SGD, we use Opacus implementation [Yousefpour et al., 2021].

**Models:** We use ResNet-18 CNN [He et al., 2016] as the base model architecture with the output of the final average pool layer as the feature embedding ($Z \in \mathbb{R}^{512}$) and the fully connected layer as the classifier. Note that for our black-box setting, we assume access to the lowest level gradient for training, but do not modify the pre-trained model in any way. For simulating the adversarial classifier, we use a 4-layer multi-layer perceptron (MLP) with 512, 1024, 256 and 4 neurons in each layer and ReLU as the activation function. As previously noted, the transformer model is a vector of the same size as the input image ($224 \times 224 \times 3$) which is added to the input (3.1). Finally, for SPAct we use the UNet [Ronneberger et al., 2015] for the image anonymizer as used in the original paper. We also use UNet as the image anonymizer for Obfuscator for a straightforward comparison. Table 3.1 shows the number of trainable parameters for each model used (excluding the simulated adversary).

Table 3.1: Comparison of Trainable Parameters

| Model | No. Trainable Parameters |
|---|---|
| UNet (SPAct, Obfuscator) | $17,267k$ |
| Cloak | $301k$ |
| **Reprogram (Ours)** | $151k$ |

The number of trainable parameters for each black-box protection method. Note that the reprogramming method only needs the same number of parameters as the input size ($224 \times 224 \times 3 \approx 151k$) and Cloak requires two times that. The UNet model size is independent of the input size.

**Evaluation Metric:** Since the distribution of the attributes is not fully balanced across the dataset, we evaluate the performance of utility and attribute inference with Average Precision and report the mean Average Precision (mAP) across labels. Note that the reported privacy values are the attribute inference of the adversary. Therefore, lower values indicate better privacy protection and are more desirable. Additionally, we report the average training time per epoch to highlight the computational complexity of each method.

**Experiments:** For the methods with a single privacy-utility trade-off hyperparameter $\lambda$ (Ours, Adversarial, SPAct and Obfuscator), we evaluate them with values of $\lambda \in \{0.3, 0.4, 0.5, 0.8\}$. For DP, we use epsilon values of $\epsilon \in \{0.1, 0.2, 0.5, 1\}$ to provide high privacy. For Cloak, we use $\gamma = 0.2$ with suppression ratios of $sr \in \{0.7, 0.8, 0.9, 0.95\}$. We set $k = 1$ for consistency across other works that train the adversary for one epoch. We implement our algorithms with PyTorch [Paszke et al., 2017] and train on a server with NVIDIA GTX 1080ti GPU and Intel i7-7700 CPU.

We train all model using Adam optimizer [Kingma and Ba, 2014] with learning rate of $10^{-3}$ with 1-Cycle Scheduling [Smith and Topin, 2019] and weight decay of $2 \cdot 10^{-5}$. For the black-box methods, we first train the black-box model normally (without privacy protection) on the $\mathcal{D}_p$ dataset. We then freeze the trained weights and use them in the other methods. For non-black-box methods, we skip this step and directly train the whole model with privacy protection from scratch. After training, we freeze all weights and train a new adversary on the feature outputs to evaluate the inference attack protection of the final model. Fig. 3.3 shows the utility-privacy trade-off between our approach and the baselines and Fig. 3.4 shows the impact of $\lambda$ on the utility-privacy trade-off for each label for our approach and Obfuscator. Additionally, Table 3.2 shows the value comparison between the methods at a similar level of utility.

Table 3.2: Experiment Results

| Method | Utility mAP (%) | Privacy mAP (%) |
|---|---|---|
| No Protection (Black-Box Model) | 89.39 | 85.32 |
| Downsample-2× | 81.66 (↓7.7) | 82.58 (↓2.7) |
| Downsample-4× | 44.97 (↓44.4) | 60.86 (↓24.5) |
| DP (Non Black-Box) ($\epsilon = 2$) | 74.04 (↓15.4) | 76.62 (↓8.7) |
| Contrastive Loss (UNet) ($\lambda = .3$) | 68.82 (↓20.6) | 77.80 (↓7.5) |
| Obfuscator (UNet) ($\lambda = .8$) | 73.34 (↓16.1) | **59.23** (↓26.1) |
| Cloak ($sr = .7$) | 72.05 (↓17.3) | 74.04 (↓11.3) |
| **Reprogram (Ours)** ($\lambda = .8$) | 74.64 (↓14.8) | **60.75** (↓24.6) |

Comparison of methods at a similar utility level where possible. Our approach outperforms other methods with the exception of Obfuscator. See Section 3.4 for more details. The number in parentheses indicate the change of accuracy w.r.t. the baseline, with the green color indicating good changes (e.g., decrease in attack accuracy) and red indicating bad changes (e.g., decrease in task accuracy).

Table 3.3: Comparison of Avg. Train Time per Epoch

| Method | Avg. Time per Epoch (s) | No. of Epochs | Total Time (m) |
|---|---|---|---|
| DP (Non Black-Box) | 550 | 50 | 458 |
| Contrastive Loss (UNet) | 3,949 | 50 | 3,291 |
| Obfuscator (UNet) | 713 | 50 | 594 |
| Cloak | 95 | 90 | 143 |
| **Reprogram (Ours)** | **82** | 50 | **68** |

Time comparison between different methods. The table shows the average runtime for one epoch, the number of epochs and the total time for each method. Note that Cloak has two training phases, and the reported time is over both training phases.

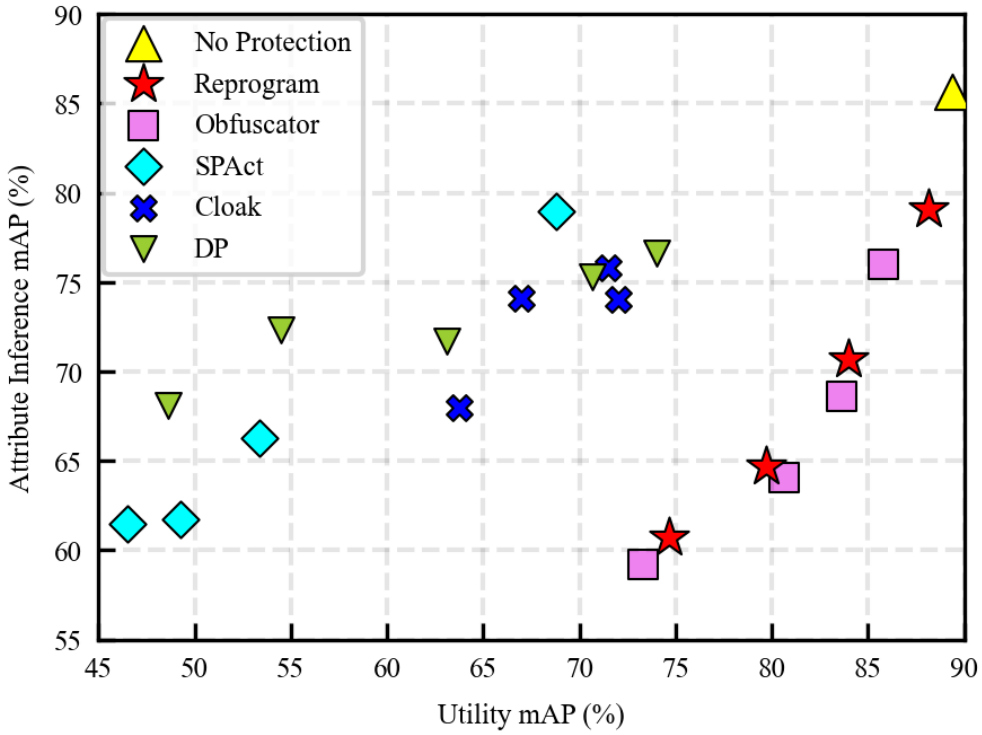
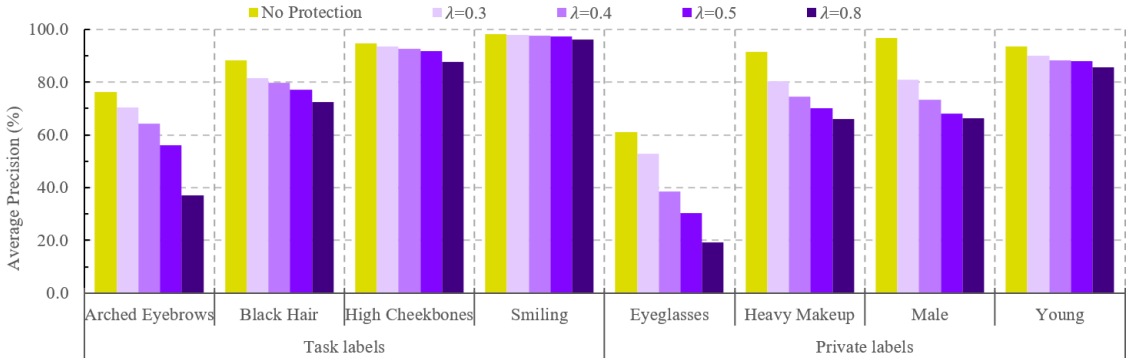



Figure 3.3: Utility and Privacy Trade-off

Privacy (y-axis) and Utility (x-axis) trade-off with different hyperparameters. Higher values in utility and lower values in privacy are desirable. The best performance would be toward the lower-right section of the plot.


(a) Our Approach




(b) Obfuscator

Figure 3.4: Utility and Privacy Per Label

Impact of $\lambda$ on utility-privacy trade-off for our method (top) and Obfuscator (bottom). The first four labels are the task labels and the last four labels are the private labels that the adversary attempts to infer. In both figures, the first bar (yellow bar) is the performance of the black-box model shown for easier comparison.

## 3.3 Ablation Study

**K:** As noted in Section 2.1.3, our algorithm (Algorithm 1) trains the adversary for $k$ steps before training the feature extractor and classifier. The reason for this is that training the adversary for multiple steps can yield a more robust adversary which in turn is more beneficial for training a privacy preserving feature extractor.

The $k$ hyperparameter provides a controllable trade-off between strengthening the adversary and the training time required. To investigate the effect of $k$ for Model Reprogramming in providing protection, we evaluate our method for $k \in \{1, 2, 4, 8\}$ and $\lambda \in \{0.3, 0.4, 0.5, 0.8\}$. The results are shown in Fig. 3.5 in form of an scatter plot. As seen in this figure, the hyperparameter $k$ does not provide any notable difference for our method. In other words, our method is able to obtain the optimal results with $k = 1$ and higher values of $k$ do not provide any benefits.

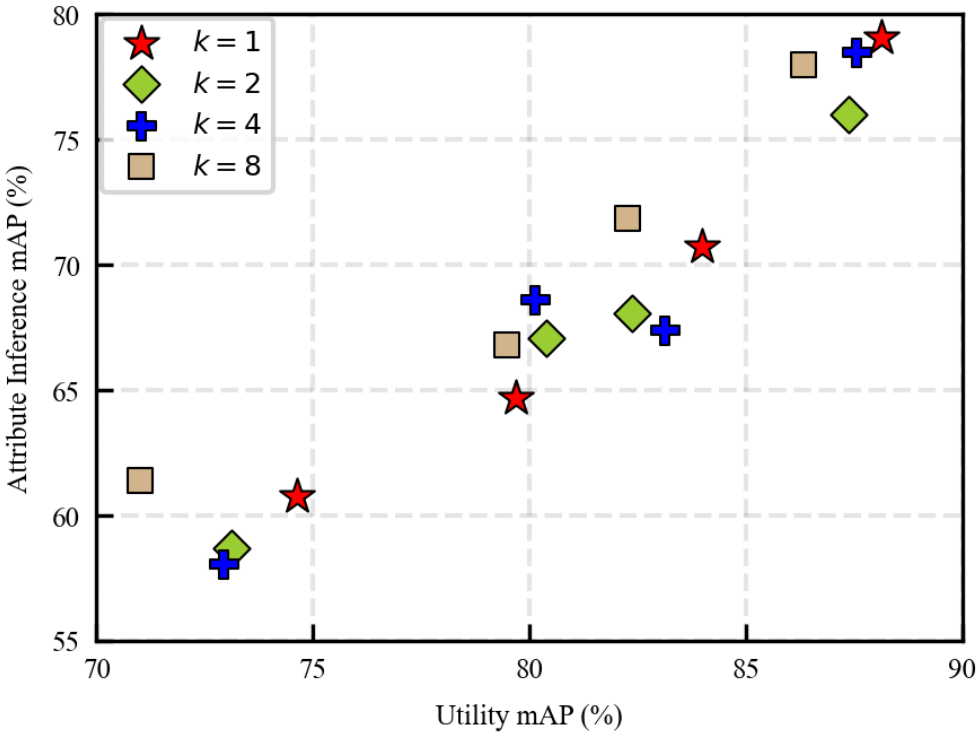



Figure 3.5: $k$ Hyperparameter Ablation Study

Privacy (y-axis) and Utility (x-axis) trade-off with different hyperparameters. As seen in the plot, tests with different values of $k$ result in a similar utility privacy trade-off.

**Logistic Function:** As described in Section 3.1, our image transformer is modeled as $x+\tanh(\delta)$ to ensure that the reprogram values fall within the range of $(-1, 1)$. To investigate the impact of this approach, we compare it with two alternative methods: one defines the image transformer as $\tanh(x+\delta)$, encompassing both the image input and added weights in a hyperbolic tangent function to restrict the output within $(-1, 1)$, and the other method simply defines the transformer as $x + \delta$, allowing unbounded outputs. To evaluate these approaches, we conducted experiments in the same setting as described in Section 3.2, using $\lambda \in \{0.3, 0.4, 0.5, 0.8\}$, and present the results as a scatter plot in Fig. 3.6. Surprisingly, despite the different approaches, the results are remarkably close and follow the same trend line, indicating that bounding the transformer output does not significantly impact the performance of the final model in our experiments.

## 3.4 Discussion

As we see in Fig. 3.3, our method constantly outperforms Cloak, DP and SPAct. That is, at a similar level of privacy protection, our method has a higher utility (towards the right side of the plot), and at a similar utility, it provides more privacy protection (towards the bottom of the plot). We suggest that this is mainly because neither DP, Cloak or SPAct specifically optimize for attribute inference protection; rather, they provide a different form of privacy that is not always effective in protecting against attribute inference attacks.

As for Obfuscator, we see our approach performs similarly. As seen in Fig. 3.3, both Obfuscator (squares) and our method (stars) follow a similar trend in the utility-

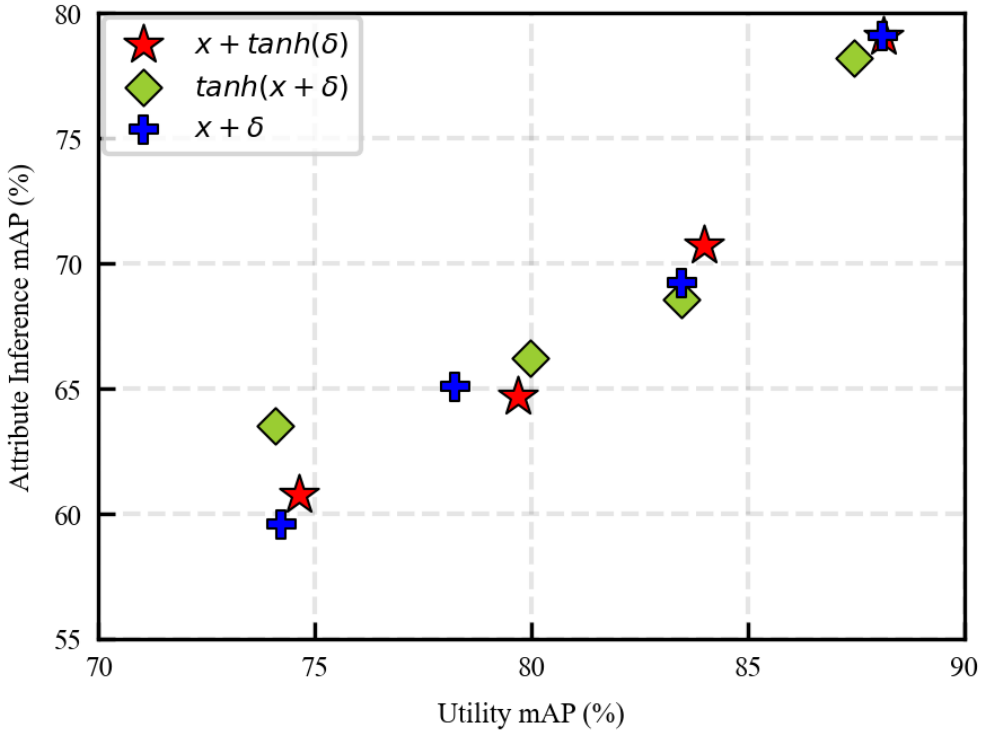



Figure 3.6: Image Transformer Ablation Study

Privacy (y-axis) and Utility (x-axis) trade-off with different hyperparameters. As seen in the plot, all three methods follow a similar trend in utility and privacy trade-off. This shows that bounding the transformer output does not greatly impact the overall performance.

privacy trade-off. However, we note that the Obfuscator's model (UNet) has more than 17 million trainable parameters, whereas reprogramming needs only about 151 thousand, two orders of magnitude less, parameters. In addition, UNet is compromised of many convolution and de-convolution operations, whereas reprogramming is only a logistic function ($tanh$) with an addition operation. While UNet is undoubtedly more powerful overall, it does not perform significantly better than reprogramming.

The difference in complexity between methods which directly impacts the training time is evident in Table 3.3. Our method has significantly lower complexity than all

other methods, with the exception of Cloak. While Cloak is the only method to have a comparable training time, our method provides a better utility-privacy trade-off. The Obfuscator model which has a similar performance to our approach, takes nearly 9 times longer to train one epoch. Overall, our method offers comparable protection to state-of-the-art methods with far less computational complexity and storage requirements.

Fig. 3.4(a) shows the average precision of utility and attribute inference attack for each label for different values of $\lambda$ compared to training without protection. As noted previously, $\lambda$ is our privacy-utility trade-off knob, with lower values prioritizing utility and higher values prioritizing privacy. The effect of this hyperparameter is visible in Fig. 3.4(a), showing that it is possible to use $\lambda$ to achieve a desirable utility-privacy trade-off that addresses our needs. Furthermore, Fig. 3.4(a) shows our method protects attribute inference attacks for 'Eyeglasses', 'Heavy Makeup' and 'Male', but does not adequately protect against the inference of the 'Young' attribute. Meanwhile, with the exception of 'Arched Eyebrows', the model's utility suffers a minimal loss of performance. We contribute the performance loss for 'Arched Eyebrows' to the fact that the 'Arched Eyebrows' and 'Eyeglasses' are closely associated in a picture, and degrading the performance in one leads to degradation of the other one; especially since the feature extractor is pre-trained and cannot be manipulated.

To examine whether these shortcomings result from our approach or are inherent to the task, we compare our method with Obfuscator – since it is the only method with comparable performance – shown in Fig. 3.4(b). As we see in Fig. 3.4(b), Obfuscator also struggles in protecting the 'Young' attribute and shows a similar

pattern in detecting 'Arched Eyebrows' and protecting 'Eyeglasses'. This suggests that the limitation is not inherent to reprogramming, but rather the tasks.

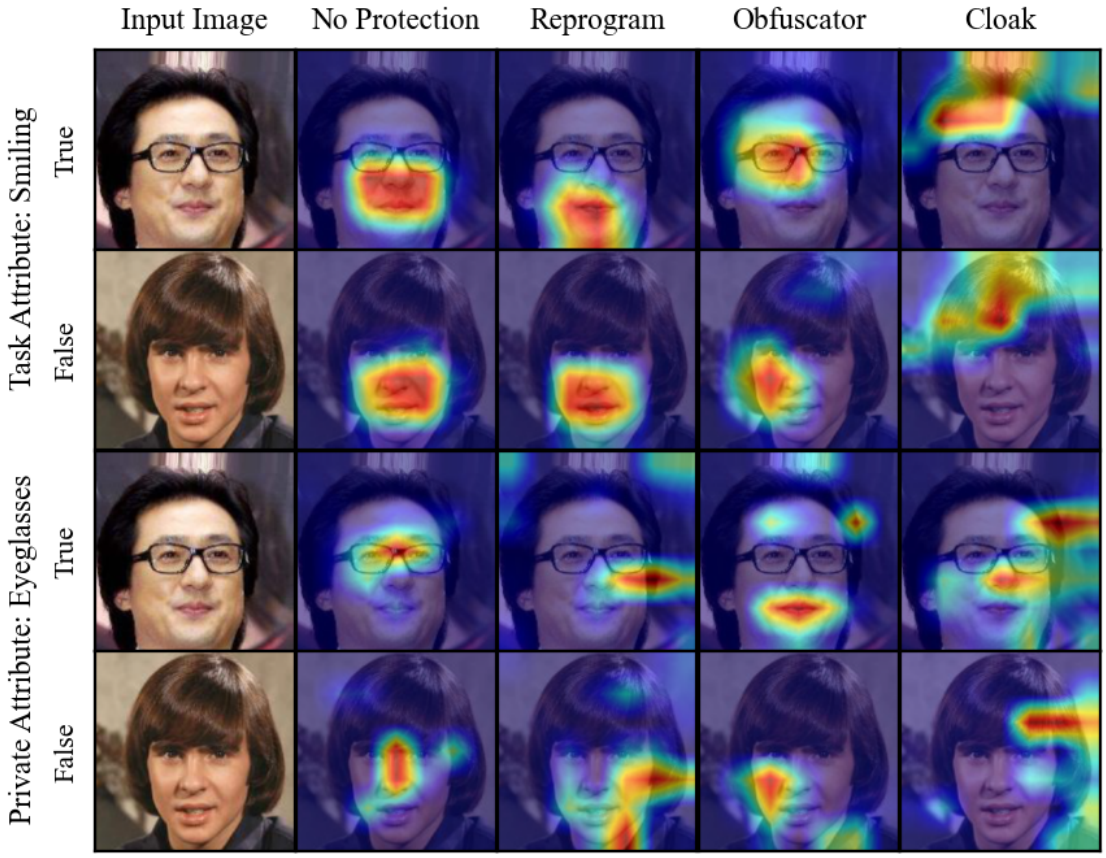



Figure 3.7: Grad-CAM analysis

Grad-CAM analysis over two sample images from CelebA [Liu et al., 2015]. The highlighted zones (marked in red) depict regions exerting a major influence on the predicted labels in each row and each protection method (columns). The first two rows are for the smiling label (Utility) and the last two rows are for the eyeglasses label (privacy). The figure shows how our method (Reprogramming) diverts the attention of the adversary for private labels while maintaining the correct attention of the utility task.

Additionally, we perform Grad-CAM [Selvaraju et al., 2017; Gildenblat and contributors, 2021] analysis to compare the performance of our method. Grad-CAM produces visual explanations showing the regions of importance in a model's predic-

tion. Fig. 3.7 shows a Grad-CAM analysis of the utility classifier on the 'Smiling' attribute and the adversary attempting to infer the 'Eyeglasses' attribute. As we see in the first two rows (task attribute), the attention of the original model (No Protection) is primarily on the mouth. Our approach maintains the general regions, while Obfuscator slightly shifts the attention, and Cloak shows the worse performance by focusing on unrelated areas of the picture. In the last two rows (private attribute), the adversary's attack on the model with no protection is heavily influenced by the eyes and the nasal bridge in the picture. In our approach, as well as Obfuscator and Cloak, we see the adversary's attention is diverted to different and unrelated regions of the input, making it harder for the adversary to infer private attributes. Overall, we see that our approach can maintain the original model's attention area for the utility task while diverting the adversary's attention to unrelated regions to maintain privacy.

## 3.5 Summary

In this chapter, we investigated the effectiveness of our proposed method for providing attribute inference protection on black-box models using Model Reprogramming. By operating at the input level, Model Reprogramming offers a promising approach for modifying the behavior of black-box models, enabling it to protect against attribute inference attacks. Through experiments on CelebA dataset, we showed that our method provides better or comparable performance compared to the state-of-the-art method while requiring significantly less computational overhead for training.

# Chapter 4

# White-Box Protection With LoRA

As the use of ML gains more traction, privacy concerns over the use of ML models and sharing features are becoming more important for designers to address. Despite the advancements in privacy, the prevalent methods for providing privacy are mostly based on complete retraining or extensively fine-tuning the model. The high costs of these methods create a significant barrier to providing privacy-conscious models for smaller organizations. In this chapter, we offer a light-weight fine-tuning method that can provide privacy to a pre-trained model with significantly less training costs. More specifically, we use a Low-Rank Adaptation (LoRA) – a popular fine-tuning method for Large Language Models (LLMs) – adapted for convolutional neural networks (CNNs), which we use to add attribute inference protection for a pre-trained model in a white-box setting. A light-weight fine-tuning method allows for robust adaptation of large pre-trained models without the costs of training from scratch or extensive fine-tuning. Additionally, LoRA provides the ability to fine-tune for different tasks using the same pre-trained model and switching to each required downstream

task as necessary.

## 4.1 Method

### 4.1.1 LoRA Model

The original LoRA fine-tuning scheme proposed by Hu et al. [2021] is specific for 2D matrices of LLMs. That is, for a weight matrix $W_0 \in \mathbb{R}^{a \times b}$, two new trainable matrices $A \in \mathbb{R}^{a \times r}$ and $B \in \mathbb{R}^{r \times b}$ are created, where $r$ is the rank of the new matrices. The rank is a hyperparameter that controls the size of the new matrices, controlling the number of new trainable parameters. Effectively, $BA$ represent $\Delta W \in \mathbb{R}^{a \times b}$ which is the update values we would add to $W_0$ if we were fine-tuning $W_0$ itself. In other words, the hypothetical updated matrix $W$ is:

$$W = W_0 + \Delta W = W_0 + BA \tag{4.1}$$

In turn, we can use (4.1) to rewrite the forward pass as:

$$h = Wx = W_0x + \Delta Wx = W_0x + BAx \tag{4.2}$$

Note that Hu et al.'s schema is specific to 2D matrices as is reliant on the fact that $B$ and $A$ are a low-rank decomposition of $\Delta W$. While 2D-CNN's weights are 4 dimensional, a similar idea is proposed by Jaderberg et al. [2014] in which they approximate the convolutional layer with a sequence of two smaller convolutional layers. Jaderberg et al.'s original objective is to enhance the efficiency of CNNs by

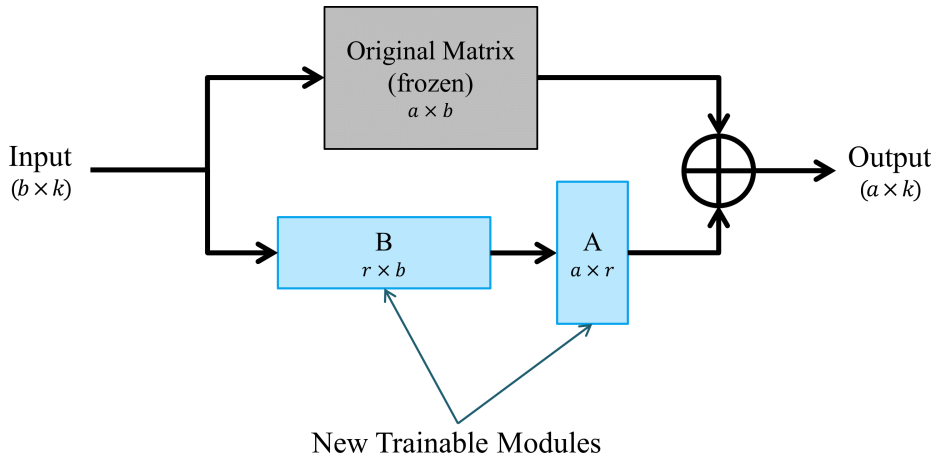



Figure 4.1: LoRA on a 2D Matrix

Illustration of Low-Rank Adaptation proposed by Hu et al. [2021]. In the top we have the original matrix of size $a \times b$ which the input goes through as usual. In addition to that, two smaller matrices of rank $r$ ($r \times b$ and $a \times r$ respectively) added in parallel, in a manner that the the resulting output has the same shape as the original output. The two outputs are added together the compromise the final output.

substituting the convolutional layers with a computationally less complex approximation. Although more complex, This approximation method is inherently very similar to (4.1) in which it can be simulated using two smaller CNNs sequentially. For a comprehensive understanding of the mathematics and further information, we refer to [Jaderberg et al., 2014] as the detailed mathematical analysis and specifics exceed the scope of this chapter.

### 4.1.2 Our Proposed Method

We adapt Jaderberg et al.'s approximation approach with Hu et al. [2021]'s fine-tuning schema, and propose the following Low-Rank Adaptation schema for CNNs: For a pre-trained Convolutional layer with $c_{out}$ filters, kernel size of $k \times k$, padding $p \times p$ and stride $s \times s$ define two smaller Convolutional layers with rank $r$ as specified

in Table 4.1 and calculate the new output with:

$$h = Conv2D_{PreTrained}(x) + (Conv2D_B \circ Conv2D_A)(x) \tag{4.3}$$

Where $\circ$ is the function composition operation. Essentially, we replace the matrix operations in (4.2) with CNN operation while ensuring the output of the new layers match the original CNN (See Fig. 4.2). The details of the new convolutional layers are shown in Table 4.1.

Similar to (4.1), the new convolutional layers $A$ and $B$ are decomposition of a virtual $\Delta W$ which is added to the original layer. With the weight of the original layer $W_0 \in \mathbb{R}^{C_{out} \times c_{in} \times k \times k}$ (where $c_{in}$ is the number of input filters), weight of the new layers are $W^A \in \mathbb{R}^{C_{out} \times r \times 1 \times k}$ and $W^B \in \mathbb{R}^{r \times c_{in} \times k \times 1}$. $\Delta W$ is then calculated as:

$$\Delta W_{j,l,m,n} = \sum_{i=0}^{r-1} W^B_{i,l,m,0} \times W^A_{j,i,0,n} \tag{4.4}$$

Table 4.1: Convolutional Layer Low-Rank Adaptation

| Convolutional Layer | No. Filters | Kernel Size | Padding | Stride |
|---|---|---|---|---|
| Pre-trained | $c_{out}$ | $k \times k$ | $p \times p$ | $s \times s$ |
| A | $c_{out}$ | $1 \times k$ | $0 \times p$ | $1 \times s$ |
| B | $r$ | $k \times 1$ | $p \times 0$ | $s \times 1$ |

The new convolutional layers $A$ and $B$ are specified in a manner that their composition ($Conv2D_B \circ Conv2D_A$) has the same output shape as the pre-trained layer. In our approach $r$ controls the intermediate number of filters between the new layers, similarly controlling the number of new trainable parameters.

As noted in Section 2.1.3, we use the adversarial Min-Max framework for training the model. We denote the Min-Max loss as $\mathcal{L}_{LoRA}$ and the feature extractor as $f_{F'}(\cdot; \theta_{F'})$ which is the modified model as explained in the previous section.

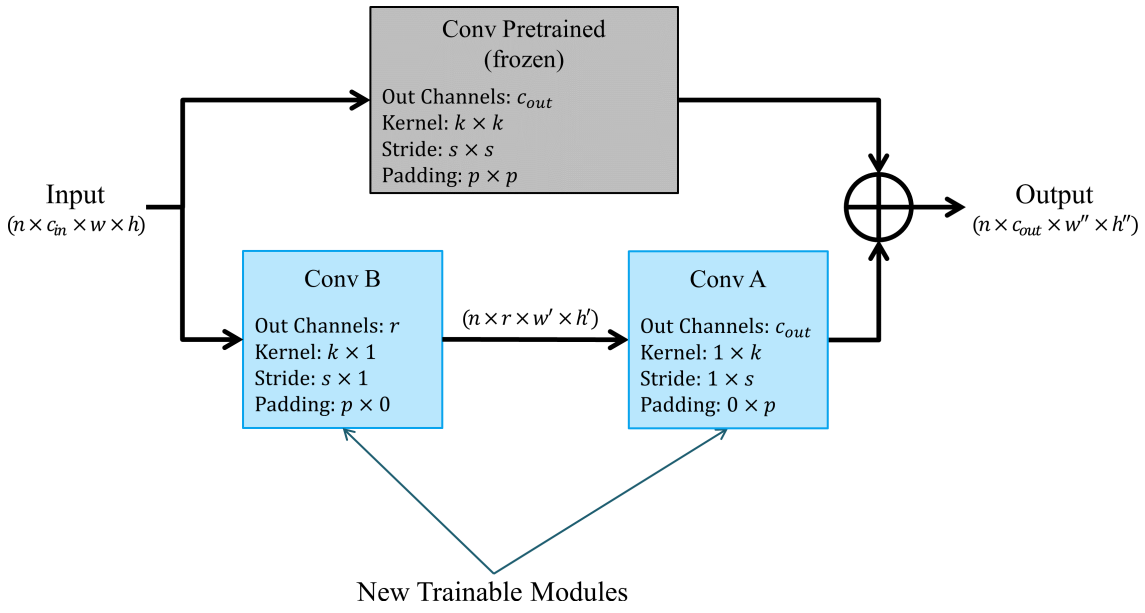



Figure 4.2: LoRA on a 2D Convolutional Layer

Our proposed LoRA schema for convolutional layers translates the benefits of [Hu et al., 2021] for 2D matrices to 2D convolutional layers.

## 4.2 Experiments

To evaluate the performance of our proposed method, we aim to compare it with the most relevant work, P-FEAT [Ding et al., 2022]. Unfortunately, the source code for P-FEAT is not available and we were unable to reproduce its results from the paper. Therefore, we aim to compare directly with the numbers reported in the paper by recreating their experimental setting.

Ding et al. evaluate their method in a min-max training framework against two types of adversaries. The first, similar to our previous chapter, is an adversary classifier that aims to detect a private attribute from the extracted features. For the second adversary, they use a reconstruction model that aims to reconstruct the original image from the extracted features. Ding et al. present this method as an unsupervised

training for defending against attribute inference attacks. We follow their experiments and evaluate our method in the same fashion.

**Datasets:** Following [Ding et al., 2022], we use the CelebA [Liu et al., 2015] dataset, containing 160K training, 20k test and 20k validation images with 40 binary facial attributes. The task is defined as classifying the *Male* attribute, while *Smile*, *Attractive*, *Mouth_Slightly_Open* and *Wearing_Lipstick* are considered private attributes. It should be noted that each private attributes is tested separately, i.e. the model has only one private label in each test. All reported results are separated by the private attribute used in the test.

We use the training set for pre-training the model and then providing defence with LoRA. To train the final adversary, we use the validation set as it is unseen data to the model. Finally, we use the test set for all evaluations reported.

**Models:** The base model is AlexNet [Krizhevsky et al., 2012] with the first five convolutional layers as the feature extractor and the final three fully connected layers as the classifier. The classifier architecture is used for both the task classifier and the adversarial classifier. After pre-training the model, we augment all convolutional layers in the feature extractor as described in Section 4.1.2 and freeze the pre-trained convolutional layers. Using $r = 1$, our final feature extractor has only 9k parameters for training, compared to original feature extractor which has 3,747k parameters. Our model has over two order of magnitude less parameters to train. For the reconstruction model we use the same architecture taken from [Dosovitskiy and Brox, 2016] and use the $L_2$ distance as the adversary's loss.

**Experiments:** We implement our algorithms with PyTorch [Paszke et al., 2017] and train on a server with NVIDIA P100 Pascal GPU and Intel E5-2650 v4 Broadwell CPU. We train all model using Adam optimizer [Kingma and Ba, 2014] with learning rate of $10^{-4}$ and weight decay of $2 \cdot 10^{-5}$. Initially, we train AlexNet normally (without protection) on the training set as our baseline for 10 epochs. After that, we augment all convolutional layers in the model as described in Section 4.1.2 with $r = 1$ and freeze the pre-trained convolutional layers. The new model is then trained with min-max framework against one of the adversaries – attribute inference adversary or reconstruction adversary – for 20 epochs. In the end, to evaluate the protection of the final model, we freeze all parameters and train a adversary on the feature outputs for 20 epochs. Similar to [Ding et al., 2022] we limit our final results to $k = 1$ and report with the optimal $\lambda$ which is $\lambda = 0.8$. The results of our experiments and its comparison to P-FEAT are shown in Table 4.2 for attribute inference adversary and in Table 4.3 for the reconstruction adversary.

Table 4.2: Experiment Results For Attribute Inference Adversary

| | No Defense (Baseline) | | P-FEAT | | LoRA [$\lambda = 0.6$] | |
|---|---|---|---|---|---|---|
| Private Attribute | Task | Attack | Task | Attack | Task | Attack |
| Smiling | 97.86 | 82.62 | 94.93 (↓2.9) | 62.55 (↓20.1) | **96.32** (↓1.5) | **49.97** (↓32.7) |
| Attractive | 97.86 | 76.55 | 77.77 (↓20.1) | 58.93 (↓17.6) | **94.16** (↓3.7) | **55.54** (↓21.0) |
| Mouth | 97.86 | 79.9 | 92.31 (↓5.6) | 76.84 (↓3.1) | **96.60** (↓1.3) | **50.49** (↓29.4) |
| Lipstick | 97.86 | 91.4 | 91.81 (↓6.1) | 61.75 (↓29.7) | **96.10** (↓1.8) | **47.81** (↓43.6) |

Results comparison between our model and P-FEAT as reported in [Ding et al., 2022] for defending against a specific privacy attack. We see our method outperforms P-FEAT on every test. The number in parentheses indicate the change of accuracy w.r.t. the baseline, with the green color indicating good changes (e.g., decrease in attack accuracy) and red indicating bad changes (e.g., decrease in task accuracy).

Table 4.3: Experiment Results For Reconstruction Adversary

| | No Defense (Baseline) | | P-FEAT | | LoRA [$\lambda = 0.6$] | |
|---|---|---|---|---|---|---|
| Private Attribute | Task | Attack | Task | Attack | Task | Attack |
| Smiling | 97.86 | 82.62 | 90.95 (↓6.9) | 64.28 (↓18.3) | **97.32** (↓0.5) | **63.32** (↓19.3) |
| Attractive | 97.86 | 76.55 | 90.95 (↓6.9) | 72.83 (↓3.7) | **97.32** (↓0.5) | **71.36** (↓5.2) |
| Mouth | 97.86 | 79.9 | 90.95 (↓6.9) | 62.57 (↓17.3) | **97.32** (↓0.5) | **60.70** (↓19.2) |
| Lipstick | 97.86 | 91.4 | 90.95 (↓6.9) | **72.37** (↓19.0) | **97.32** (↓0.5) | 91.36 (↓0) |

Results comparison between our model and P-FEAT as reported in [Ding et al., 2022] with using reconstruction attack as an adversarial objective. Our method is able to outperform P-FEAT's by having a higher utility and slightly better privacy on all labels except *Wearing_Lipstick*. The number in parentheses indicate the change of accuracy w.r.t. the baseline, with the green color indicating good changes (e.g., decrease in attack accuracy) and red indicating bad changes (e.g., decrease in task accuracy).

## 4.3 Ablation Study

To further examine our proposed method, we conduct several ablation studies explained in this section. Specifically, we study the effect of selection of which layers to fine-tune and the hyperparameter rank ($r$) and $k$ on utility and privacy trade-off. In the interest of maintaining the ablation study's conciseness and clarity, we exclusively focuses on the private attribute of *Attractive*. This approach enables a more streamlined and straightforward examination of our method's effectiveness in preserving attribute privacy for this specific characteristic. By limiting the scope to a single attribute, we can assess the impact of various parts of the method more effectively and draw meaningful conclusions.

**Selection of Layers:** The Alexnet Model [Krizhevsky et al., 2012] is compromised of five sequential CNN layers followed by three fully connected layers which we denote

as $CONV1 \rightarrow CONV2 \rightarrow CONV3 \rightarrow CONV4 \rightarrow CONV5 \rightarrow FC6 \rightarrow FC7 \rightarrow FC8$. As explained in Section 4.2, the convolutional layers constitute our feature extractor which we fine-tune using our proposed method. While in our experiments we fine-tune all of there convolutional layers with our proposed method, it is possible to only fine-tune select layers, further lower fine-tuning costs for our tasks. To evaluate how fine-tuning each layer impacts the models performance, we experiment with fine-tuning each layer individually, and compare it with fine-tuning all layers For there tests, we fix $r = 1$, $k = 1$ and test with $\lambda \in \{0.5, 0.6, 0.7, 0.8\}$. The results are shown in Fig. 4.3. As we see in the figure, fine-tuning the later layers provides a better privacy-utility trade-off – with the exception of the last layer – and fine-tuning all layers provides the best performance. Interestingly, we see fine-tuning the final layer (CONV5) does not provide a better performance and does not follow a similar trend-line to the other tests.

**Rank:** As explained in Section 4.1, the rank hyperparameter ($r$) controls the the size of the new convolutional layers added to the model. To evaluate how the rank impacts the performance of our model, we repeat the experiments with ranks $r \in \{1, 4, 16, 64\}$ and $\lambda \in \{0.5, 0.6, 0.7, 0.8\}$, while setting $k = 1$ and fine-tuning all the convolutional layers.

Fig. 4.4 shows the performance comparisons of different ranks. Surprisingly, we see increasing the rank does not increase the overall performance, but rather degrades the utility-privacy trade-off to the point where models with $r = 16$ and $r = 64$ fail to provide any notable privacy protection. We believe this due to the fact that in our experimental settings, the lower number of parameters is sufficient and increasing the

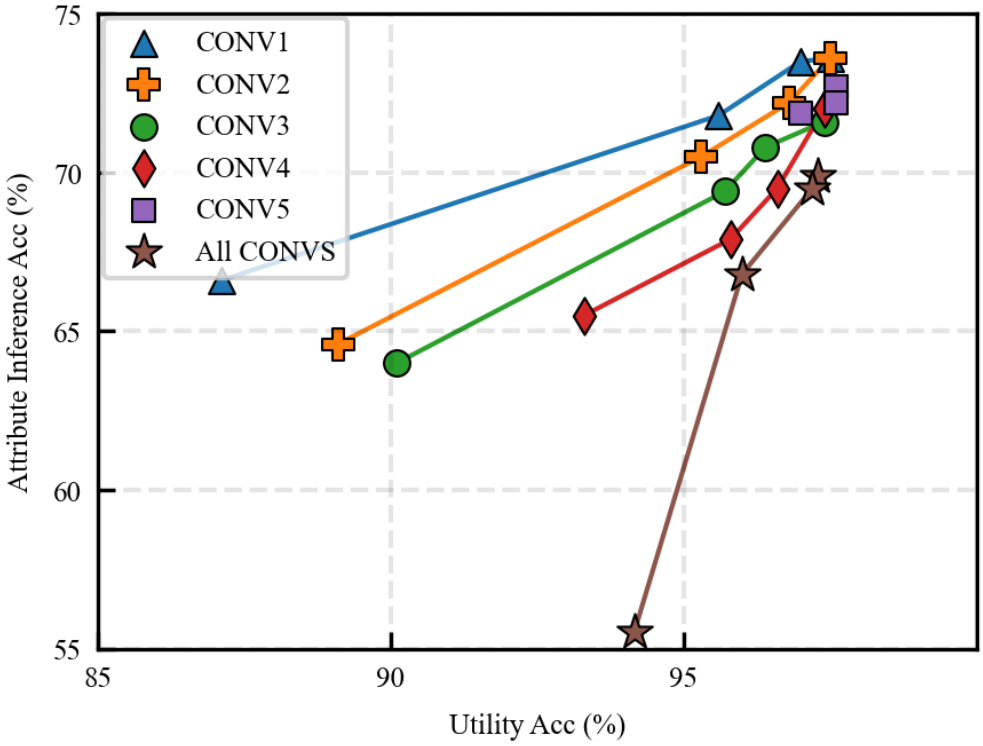



Figure 4.3: LoRA Fine-Tuning Layer Ablation Study

Attack accuracy (y-axis) and Utility accuracy (x-axis) trade-off with fine tuning different layer. Higher values in utility and lower values in privacy are desirable. The best performance would be toward the lower-right section of the plot. For clarity, test points with the same layers are connected with a line of same color. The figure shows that fine-tuning later layers has a stronger impact and a better privacy-utility trade-off compared to the earlier layers, with the exception of the fifth layer. Fine-tuning all layers also has a stronger impact, more visible for higher values of $\lambda$.

size makes the training process more difficult. More specifically, since we pre-train the model, our fine-tuning process is mainly focused on adding privacy and managing trade-off with the existing utility; therefore, our experiment does not benefit from the increase in rank.

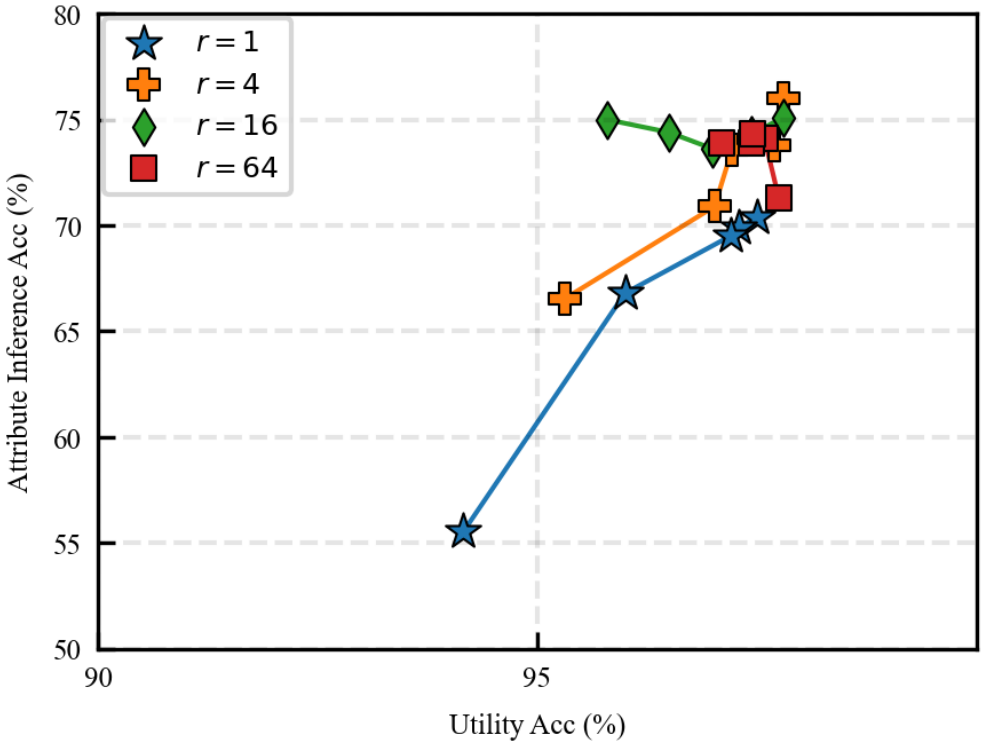



Figure 4.4: LoRA Fine-Tuning Rank Ablation Study

Attack accuracy (y-axis) and Utility accuracy (x-axis) trade-off with fine tuning different layer. Higher values in utility and lower values in privacy are desirable. The best performance would be toward the lower-right section of the plot. For clarity, test points with the same rank are connected with a line of same color. The figure shows that increasing the rank has an adverse effect on performance for our experiments. We attribute this behavior to the settings of our specific experiment, rather than a limitation of the fine-tuning method itself.

**K:** Our Min-Max framework algorithm (Algorithm 1) includes a hyperparameter $k$, denoting the number of steps the adversary is trained before we train the feature extractor and classifier. The intuition behind $k$ is that a stronger adversary is more helpful in guiding our model to attain protection and $k$ is the hyperparameter that provides the trade-off between strengthening the adversary and the training time required. To investigate how much this hyperparameter effects our proposed method

we experiment $k \in \{1, 2, 4\}$ for $\lambda \in \{0.5, 0.6, 0.7, 0.8\}$, with $r = 1$ and fine-tuning all the convolutional layers. The results of this study are show in Fig. 4.5. As we see in the figure, tests with $k = 1$ and $k = 2$ perform similarly, while increasing $k = 4$ results in a worse utility-privacy trade-off. We believe this because increasing the $k$ value focuses more of the training resources for the adversary and less for the feature extractor model and therefore increasing $k$ beyond a certain threshold proves detrimental to the training process.

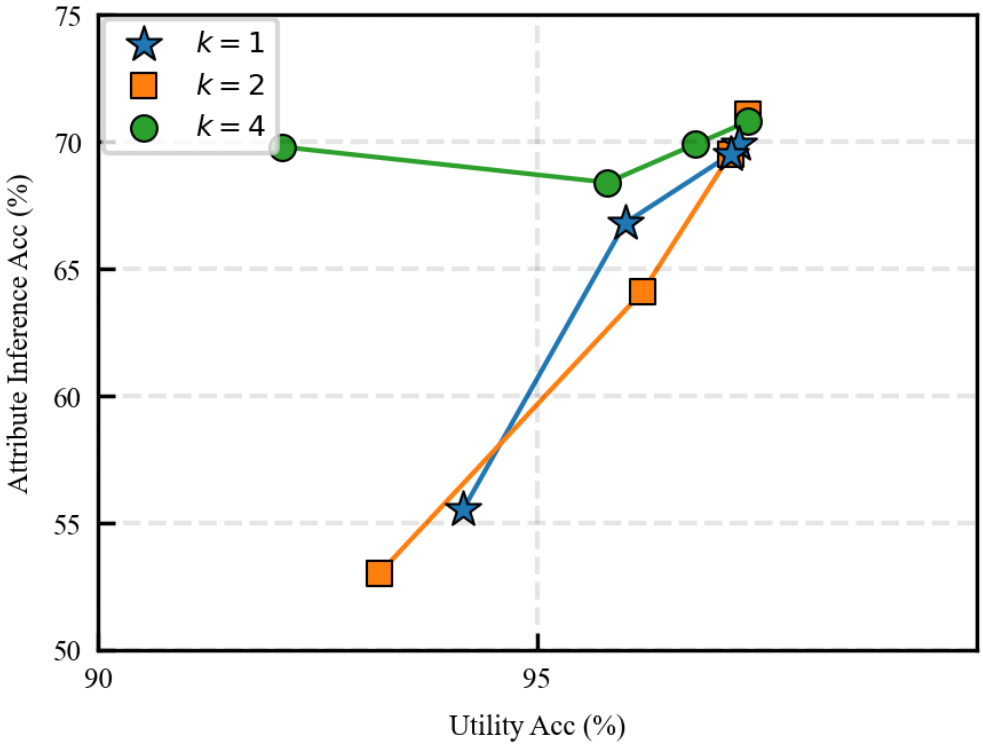



Figure 4.5: LoRA Fine-Tuning K Ablation Study

Attack accuracy (y-axis) and Utility accuracy (x-axis) trade-off with fine tuning different layer. Higher values in utility and lower values in privacy are desirable. The best performance would be toward the lower-right section of the plot. For clarity, test points with the same $k$ are connected with a line of same color. The figure shows increasing $k$ beyond 2 deteriorates the performance of the final model.

## 4.4 Discussion

As we see in Table 4.2, our method constantly outperforms P-FEAT over all tests with attribute inference adversary, offering both better task accuracy and better privacy. Note that the attack accuracy of 50% indicates the adversary is randomly guessing which is the best possible outcome. Our method is able to achieve this over three of the four labels. As for the reconstruction adversary, Table 4.3 show our method outperforms P-FEAT with the exception of *Wearing_Lipstick* attribute, on which our method is unable to provide adequate protection. We suggest two primary reasons for this. Firstly, using a reconstruction adversary, while beneficial is not enough to fully protect sensitive attributes. Secondly, the task attribute, *Male*, has a strong correlation with the private attribute, *Wearing_Lipstick* as shown in Table 4.4. The shortcomings of the reconstruction adversary is also visible in regards to other attributes, showing a notably lower protection (more than 10%) compared to the attribute inference adversary. We believe this in combination with the high correlation between the attributes results in our model being unable to provide adequate protection for the *Wearing_Lipstick* attribute.

Table 4.4: Correlation Between Selected Attributes

| | Male | Attractive | Mouth_Slightly_Open | Smiling | Wearing_Lipstick |
|---|---|---|---|---|---|
| Male | 1.00 | -0.4 | -0.1 | -0.14 | -0.79 |

The correlation between the utility attribute (*Male*) and private attributes reveals an important feature. The *Male* attribute has a high correlation with the *Wearing_Lipstick* private attribute. This characteristic of the chosen attribute means that hiding the *Wearing_Lipstick* attribute while keeping the *Male* attribute is a much more difficult task compared to other attributes.

## 4.5 Summary

In this chapter, we explored the application of Low-Rank Adaptation (LoRA) in the white-box setting for providing attribute inference protection in pre-trained convolutional neural networks (CNNs). We demonstrated that LoRA as an fine-tuning technique is applicable in CNN architectures and furthermore it is capable of providing superior utility and privacy protection compared to the state-of-the-art while requiring significantly less parameters for training.

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusions

In this thesis, we conducted a comprehensive study of two recent fine-tuning techniques, Model Reprogramming and Low-Rank Adaptation (LoRA), to address the issue of attribute inference protection for pre-trained models. Our investigation aimed to bridge the gap between efficient transfer learning and robust privacy protection, offering practical solutions for secure and privacy-conscious machine learning services.

We successfully demonstrated the potential of Model Reprogramming as an effective method for providing protection for pre-trained models in the black-box setting. By operating at the input level, Model Reprogramming is able to manipulate the model's output to conceal sensitive attributes without the need to modify the original model. We've shown that our approach can perform as good as existing methods in providing attribute inference protection with considerably less computational complexity.

Moreover, our extension of Low-Rank Adaptation tailored for convolutional neural networks showed notable improvement in providing protection compared to the most recent work in attribute inference protection. This novel approach enables fine-tuning comparable to established methods utilized by Large Language Models, making it an attractive option for implementing privacy-preserving fine-tuning on pre-trained vision models.

## 5.2 Future Work

It is important to acknowledge that despite the promising results, further evaluation is necessary to assess the robustness of our proposed methods under different settings and practicality in real-world applications.

Future research could significantly benefit from exploring the broader applicability of our proposed methods in addressing various privacy aspects, including but not limited to membership inference, reconstruction, and model-inversion attacks. We believe future studies can provide us with a better understanding of the robustness and limitations of our techniques for providing inexpensive privacy protection in ML models. We also consider the potential of our approach for use across different datasets and domains to be promising for future research.

# Bibliography

M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep Learning with Differential Privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016. doi: 10.1145/2976749.2978318.

H. Bahng, A. Jahanian, S. Sankaranarayanan, and P. Isola. Exploring visual prompts for adapting large-scale models. *ArXiv*, abs/2203.17274, 2022.

J. Chang and J. Sha. An efficient implementation of 2D convolution in CNN. *IEICE Electronics Express*, 14(1), 2017. doi: 10.1587/elex.13.20161134.

P.-Y. Chen. Model reprogramming: Resource-efficient cross-domain machine learning. *ArXiv*, abs/2202.10629, 2022.

I. R. Dave, C. Chen, and M. Shah. SPAct: Self-Supervised Privacy Preservation for Action Recognition. In *CVPR*, 2022. doi: 10.1109/CVPR52688.2022.01953.

X. Ding, H. Fang, Z. Zhang, K.-K. R. Choo, and H. Jin. Privacy-Preserving Feature Extraction via Adversarial Training. *"IEEE Trans. Knowl. Data Eng."*, 34(4), 2022. ISSN 1558-2191. doi: 10.1109/TKDE.2020.2997604.

A. Dosovitskiy and T. Brox. Inverting Visual Representations With Convolutional Networks. In *CVPR*, pages 4829–4837, 2016.

C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4), 2014. ISSN 1551-305X, 1551-3068. doi: 10.1561/0400000042.

G. F. Elsayed, I. J. Goodfellow, and J. N. Sohl-Dickstein. Adversarial reprogramming of neural networks. *ArXiv*, abs/1806.11146, 2018.

J. Gildenblat and contributors. Pytorch library for cam methods. `https://github.com/jacobgil/pytorch-grad-cam`, 2021.

X. Guo, B. Li, and H. Yu. Improving the Sample Efficiency of Prompt Tuning with Domain Adaptation. *ArXiv*, abs/2303.02861, 2022.

Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris. SpotTune: Transfer Learning Through Adaptive Fine-Tuning. In *CVPR*, 2019.

K. Hambardzumyan, H. Khachatrian, and J. May. Warp: Word-level adversarial reprogramming. In *ACL-IJCNLP*, 2021. doi: 10.18653/v1/2021.acl-long.381.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv*, abs/2106.09685, 2021.

H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang. Membership Inference Attacks on Machine Learning: A Survey. *ACM Computing Surveys*, 2022. ISSN 0360-0300. doi: 10.1145/3523273.

M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up Convolutional Neural Networks with Low Rank Expansions. *ArXiv*, abs/1405.3866, 2014.

M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim. Visual Prompt Tuning. In *ECCV*, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-19826-7. doi: 10.1007/978-3-031-19827-4_41.

M. Kim, H. Kim, and Y. M. Ro. Speaker-adaptive lip reading with user-dependent padding. *ArXiv*, abs/2208.04498, 2022.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv*, abs/1412.6980, 2014.

W. W. Koczkodaj, M. Mazurek, D. Strzałka, A. Wolny-Dominiak, and M. Woodbury-Smith. Electronic Health Record Breaches as Social Indicators. *Social Indicators Research*, 141(2), 2019. doi: 10.1007/s11205-018-1837-z.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*, volume 25. Curran Associates, Inc., 2012.

A. Li, Y. Duan, H. Yang, Y. Chen, and J. Yang. TIPRDC: Task-Independent Privacy-Respecting Data Crowdsourcing Framework for Deep Learning with Anonymized Intermediate Representations. In *SIGKDD*, 2020. doi: 10.1145/3394486.3403125.

A. Li, J. Guo, H. Yang, F. D. Salim, and Y. Chen. DeepObfuscator: Obfuscating Intermediate Representations with Privacy-Preserving Adversarial Learning on Smartphones. In *IoTDI*, 2021. doi: 10.1145/3450268.3453519.

H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. Raffel. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. *ArXiv*, abs/2303.02861, 2022a.

X. Liu, H. Zhao, M. Tian, L. Sheng, J. Shao, S. Yi, J. Yan, and X. Wang. HydraPlus-Net: Attentive Deep Features for Pedestrian Analysis. In *ICCV*, 2017. doi: 10.1109/ICCV.2017.46.

Y. Liu, R. Wen, X. He, A. Salem, Z. Zhang, M. Backes, E. D. Cristofaro, M. Fritz, and Y. Zhang. ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models. In *USENIX Security*, 2022b. ISBN 978-1-939133-31-1.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *S&P*, 2019. doi: 10.1109/SP.2019.00029.

S. Meredith. Facebook-cambridge analytica: A timeline of the data hijacking scandal,

2018. URL https://www.cnbc.com/2018/04/10/facebook-cambridge-analytica-a-timeline-of-the-data-hijacking-scandal.html.

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 *WWW*, 2021. ISBN 978-1-4503-8312-7. doi: 10.1145/3442381.3449965.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

J. Pfeiffer, S. Ruder, I. Vulić, and E. M. Ponti. Modular Deep Learning. *ArXiv*, abs/2302.11529, 2023.

O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015. ISBN 978-3-319-24574-4. doi: 10.1007/978-3-319-24574-4_28.

F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. doi: 10.1109/CVPR.2015.7298682.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

L. N. Smith and N. Topin. Super-convergence: very fast training of neural networks

using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*. SPIE, 2019. doi: 10.1117/12.2520589.

C. Song and V. Shmatikov. Overlearning reveals sensitive attributes. *ArXiv*, abs/1905.11742, 2019.

M. Treviso, J.-U. Lee, T. Ji, B. van Aken, Q. Cao, M. R. Ciosici, M. Hassid, K. Heafield, S. Hooker, C. Raffel, P. H. Martins, A. F. T. Martins, J. Z. Forde, P. Milder, E. Simpson, N. Slonim, J. Dodge, E. Strubell, N. Balasubramanian, L. Derczynski, I. Gurevych, and R. Schwartz. Efficient Methods for Natural Language Processing: A Survey. *ArXiv*, abs/2209.00099, 2023.

Y.-Y. Tsai, P.-Y. Chen, and T.-Y. Ho. Transfer Learning without Knowing: Reprogramming Black-box Machine Learning Models with Scarce Data and Limited Resources. In *PMLR*, 2020.

Z. Wang, R. Panda, L. Karlinsky, R. Feris, H. Sun, and Y. Kim. Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning. *ArXiv*, abs/2303.02861, 2023.

C. Williams. 620 million accounts stolen from 16 hacked websites now for sale on dark web, seller boasts, 2019. URL `https://www.theregister.com/2019/02/11/620_million_hacked_accounts_dark_web/`.

Z. Wu, Z. Wang, Z. Wang, and H. Jin. Towards Privacy-Preserving Visual Recognition via Adversarial Training: A Pilot Study. In *ECCV*, 2018.

Z. Wu, H. Wang, Z. Wang, H. Jin, and Z. Wang. Privacy-Preserving Deep Action

Recognition: An Adversarial Learning Framework and A New Dataset. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(4), 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.3026709.

T. Xiao, Y.-H. Tsai, K. Sohn, M. Chandraker, and M.-H. Yang. Adversarial Learning of Privacy-Preserving and Task-Oriented Representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07), 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i07.6930.

A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, and I. Mironov. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*, 2021.

C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation. In *ECCV*, 2018.

D. Yu, S. Naik, A. Backurs, S. Gopi, H. A. Inan, G. Kamath, J. Kulkarni, Y. T. Lee, A. Manoel, L. Wutschitz, S. Yekhanin, and H. Zhang. Differentially Private Fine-tuning of Language Models. *ArXiv*, abs/2110.06500, 2021.

G. Zhang, Y. Zhang, Y. Zhang, W. Fan, Q. Li, S. Liu, and S. Chang. Fairness Reprogramming. In *NIPS*, 2022.

J. O. Zhang, A. Sax, A. R. Zamir, L. J. Guibas, and J. Malik. Side-tuning: Network adaptation via additive side networks. *ArXiv*, abs/1912.13503, 2019.

Y. Zheng, X. Feng, Z. Xia, X. Jiang, A. Demontis, M. Pintor, B. Biggio, and F. Roli.

Why adversarial reprogramming works, when it fails, and how to tell the difference. *ArXiv*, abs/2108.11673, 2021.

Z. Zhou, J. Shin, L. Zhang, S. Gurudu, M. Gotway, and J. Liang. Fine-Tuning Convolutional Neural Networks for Biomedical Image Analysis: Actively and Incrementally. In *CVPR*, 2017.

Y. Zhu, X. Yang, Y. Wu, and W. Zhang. Parameter-Efficient Fine-Tuning with Layer Pruning on Free-Text Sequence-to-Sequence Modeling. *ArXiv*, abs/2303.02861, 2023.