Not All Features Are Equal: Discovering Essential Features for Preserving Prediction Privacy

Fatemehsadat Mireshghallah Mohammadkazem Taram Ali Jalali[†]
Ahmed Taha Elthakeb[*] Dean Tullsen Hadi Esmaeilzadeh
{fmireshg,mtaram}@eng.ucsd.edu,ajjalali@amazon.com,ahmed.t.althakeb@gmail.com,{tullsen,hadi}@eng.ucsd.edu
University of California San Diego
[†]Amazon.com, Inc. [*]Samsung Electronics

ABSTRACT

When receiving machine learning services from the cloud, the provider does not need to receive all features; in fact, only a subset of the features are necessary for the target prediction task. Discerning this subset is the key problem of this work. We formulate this problem as a gradient-based perturbation maximization method that discovers this subset in the input feature space with respect to the functionality of the prediction model used by the provider. After identifying the subset, our framework, Cloak, suppresses the rest of the features using utility-preserving constant values that are discovered through a separate gradient-based optimization process. We show that Cloak does not necessarily require collaboration from the service provider beyond its normal service, and can be applied in scenarios where we only have black-box access to the service provider's model. We theoretically guarantee that Cloak's optimizations reduce the upper bound of the Mutual Information (MI) between the data and the sifted representations that are sent out. Experimental results show that Cloak reduces the mutual information between the input and the sifted representations by 85.01% with only negligible reduction in utility (1.42%). In addition, we show that Cloak greatly diminishes adversaries' ability to learn and infer non-conducive features.

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; **Usability in security and privacy**; • **Computing methodologies** → *Neural networks*; Computer vision tasks; • **Mathematics of computing** → Information theory.

KEYWORDS

Privacy-preserving Machine Learning, Deep Learning, Fairness

ACM Reference Format:
Fatemehsadat Mireshghallah Mohammadkazem Taram Ali Jalali[†] and Ahmed Taha Elthakeb[*] Dean Tullsen Hadi Esmaeilzadeh. 2021. Not All Features Are Equal: Discovering Essential Features for Preserving Prediction Privacy. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449965

This work is licensed under a Creative Commons Attribution International 4.0 License.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia
© 2021 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449965

1 INTRODUCTION

The computational complexity of Machine Learning (ML) models has pushed their execution to the cloud. The edge devices on the user side capture and send their data to the cloud for *prediction services*. On the one hand, this exchange of data for services has become pervasive since the provider can enhance the user experience by potentially using the data for the betterment of its services [68], which in many cases is offered for free. On the other hand, as soon as the data is sent to the cloud, it can be misused by the cloud provider, or leaked through security vulnerabilities even if the cloud provider is trusted [35, 43, 59, 80, 81]. The insight in this paper is that a large fraction of the data is not relevant to the prediction service and can be sifted prior to sending the data out, thus enabling access to the services with much greater privacy. As such, we propose Cloak, an orthogonal approach to the existing techniques that mostly rely on cryptographic solutions and impose prohibitive delays and computational cost. Table 1 summarizes most state-of-the-art encryption-based methods and their runtime compared to unencrypted execution on GPUs. As shown, these techniques impose between 318× to 14,000× slowdown. An image classification inference is performed in multiple seconds, an order of magnitude away from the service-level agreement between users and cloud providers, which is between 10 to 100 milliseconds according to MLPerf industry measures [55, 69]. Such slowdowns will lead to unacceptable interaction with services that require near real-time response (e.g., home automation cameras). Cloak provides a middle ground, where there is a provable degree of privacy while the prediction latency is essentially unaffected. To that end, Cloak only sends out the features that the provider essentially requires to carry out the requested service. Existing privacy techniques are applicable to scenarios that can tolerate longer delays, but are not currently suitable for consumer applications, which rely on interactive prediction services. However, having no privacy protection is also not desirable.

To that end, this paper presents Cloak, a framework that sifts the features of the data based on their relevance to the target prediction task. To solve this problem, we reformulate the objective as a *gradient-based* optimization problem, that generates a *sifted representation of the input*. The intuition is that if a feature can consistently tolerate the addition of noise without degrading the utility, that feature is not conducive to the classification task. As such, we augment each feature i with a scaled addition of a noise distribution ($\sigma_i.\mathcal{N}(0,1)$) and learn the scales ($\sigma_i$s). To learn the scales, we start with a pre-trained classifier with known parameters and drive a loss function with respect to the scales while the formulation comprises the model as a

Table 1: Slowdown of cryptographic techniques vs. conventional GPU execution on Titan Xp and Cloak.

Cryptographic Technique	Release Year	DNN	Dataset	Prediction Time (sec)			Slowdown
				Encrypted	Conventional	Cloak	
FALCON [85]	2020	VGG-16	ImageNet	12.96	0.0145	0.0148	906×
DELPHI [54]	2020	ResNet-32	CIFAR-100	3.5	0.0112	0.0113	318×
CrypTen [22]	2019	ResNet-18	ImageNet	8.30	0.0121	0.0123	691×
GAZELLE [30]	2018	ResNet-32	CIFAR-100	82.00	0.0112	0.0113	7,454×
MiniONN [45]	2017	LeNet-5	MNIST	9.32	0.0007	0.0007	14,121×

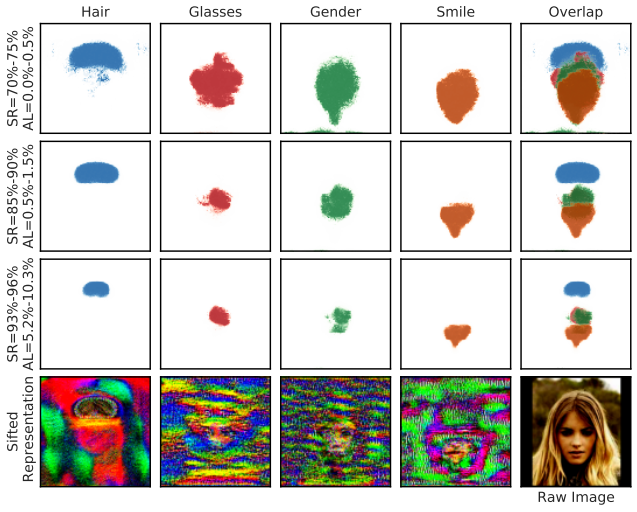


Figure 1: Cloak's discovered features for target DNN classifiers (VGG-16) for black-hair color, eyeglasses, gender, and smile detection. The colored features are conducive to the task. The 3 sets of features depicted for each task correspond to different suppression ratios (SR). AL denotes the range of accuracy loss imposed by the suppression.

known analytical function. The larger the scales, the larger the noise that can be added to a corresponding feature, and the less conducive the feature is. As such, the learned scales are thresholded to suppress the non-conducive features to a constant value, which yields the sifted representation of the input. By removing such features, Cloak guarantees that no information about them can be learned or inferred from the sifted representation that the consumer sends. Figure 1 shows examples of conducive features for multiple tasks discovered by Cloak and the corresponding sifted representation for an example image. Our differentiable formulation of finding the scales minimizes the upper bound of the Mutual Information (MI) between the irrelevant features and the sifted representation (maximizing privacy) while maximizing the lower bound of MI between the relevant features and the generated representation (preserving utility).

Experimental evaluation with real-world datasets of UTKFace [87], CIFAR-100 [37], and MNIST [40] shows that Cloak can reduce the mutual information between input images and the publicized representation by 85.01% with an accuracy loss of only 1.42%. In addition, we evaluate the protection offered by Cloak against adversaries that try to infer data properties from sifted representations on CelebA dataset [47]. We show that sifted representations generated for "smile detection" as the target task effectively prevent adversaries from inferring information about hair color and/or eyeglasses. We show that Cloak can provide these protections even in a black-box setting where we do not have access to the service provider's model parameters or architecture. Additionally, we show that Cloak outperforms Shredder [52], a recent work in prediction privacy that heuristically samples and reorders additive noise at run time to imitate the previously collected patterns. We further show that Cloak can improve the classifier's fairness. The code for the proposed method is available at https://github.com/mireshghallah/cloak-www-21, and the details of the experimental setup and the hyperparameters used for the evaluations are provided in the appendix.

2 PRELIMINARIES

In this section, we discuss the notation and fundamental concepts used in the rest of the paper, starting with our threat model.

Threat Model. We assume a remote prediction service setup, where a specific target prediction task is executed on input data. Our goal is to create a representation $\mathbf{x_s}$ of the input data $\mathbf{x}$ that has only the features that are essential to the target task, and suppresses excessive features in the input. We then send this $\mathbf{x_s}$ to the service provider. For our theoretical and empirical evaluations, we adopt supervised classification tasks as our target. We assume two access modes to the target classifier f_θ: white-box and black-box. In the white-box setup, we assume access to the architecture and parameters θ of the target classifier. In the black-box setup, we have no access to the target classifier, nor the data it was trained on. In both cases, we need labeled training data from the data distribution $\mathcal{D}$, that the target classifier was trained on. We do not, however, need access to the exact same training data, nor do we need any extra collaboration from the service provider, such as a change in infrastructure or model parameters.

Feature Space. We assume each given input $\mathbf{x}$ to be a collection of features, and group these features based on their importance for the decision making of the target classifier, f_θ. We define the two disjoint feature groups of conducive features, $\mathbf{c}$, which are those relevant to the target task and important to f_θ and non-conducive features, $\mathbf{u}$, which are less relevant. Our goal is to find the conducive features and only keep them.

Mutual Information. The amount of mutual information between the raw data $\mathbf{x}$, and the representation that is to be publicized, $\mathbf{x_s}$ is a measure of privacy that is widely used in literature [16, 33, 42],

and is denoted by $I(\mathbf{x};\mathbf{x_s})$. Cloak aims at learning representations $\mathbf{x_s}$ that decrease this mutual information while maintaining the accuracy of the target classification task. Formally, Cloak tries to minimize $I(\mathbf{x_s};\mathbf{u})$ while maximizing $I(\mathbf{x_s};\mathbf{c})$.

3 CLOAK'S OPTIMIZATION PROBLEM

This section formally describes the optimization problem and presents a computationally tractable method towards solving it. Let $\mathbf{x}\in\mathbf{R}^n$ be an input, and $\mathbf{c}\subseteq\mathbf{x}$ and $\mathbf{u}\subseteq\mathbf{x}$ be two disjoint sets of conducive and non-conducive features with respect to our target classifier (f_θ). We construct a noisy representation $\mathbf{x_c}=\mathbf{x}+\mathbf{r}$ where $\mathbf{r}\sim\mathcal{N}(\boldsymbol{\mu},\Sigma)$ and Σ is a diagonal covariance matrix, as we set the elements of the noise to be independent. This noisy representation helps find the conducive features and is used to create a final suppressed representation $\mathbf{x_s}$ that is sent to the service provider. The goal is to construct $\mathbf{x_c}$ such that the mutual information between $\mathbf{x_c}$ and $\mathbf{u}$ is minimized (for privacy), while the mutual information between $\mathbf{x_c}$ and $\mathbf{c}$ is maximized (for utility). The is written as the following soft-constrained optimization problem:

$$\min_{\mathbf{x_c}} \quad I(\mathbf{x_c};\mathbf{u})-\lambda I(\mathbf{x_c};\mathbf{c}) \tag{1}$$

The intuitive solution is to set $\mathbf{x_c}=\mathbf{c}$. But, directly finding $\mathbf{c}$ is, in most cases, not tractable due to the high complexity of classifiers. To solve this problem, we bound the terms of our optimization problem of Equation 1, and then take an iterative approach [8]. To this end, we find an upper bound for $I(\mathbf{x_c};\mathbf{u})$ and a lower bound for $I(\mathbf{x_c};\mathbf{c})$.

3.1 Upper bound on $I(\mathbf{x_c};\mathbf{u})$

Since $\mathbf{u}$ is a subset of $\mathbf{x}$, the following holds:

$$I(\mathbf{x_c};\mathbf{u})\leq I(\mathbf{x_c};\mathbf{x})=\mathcal{H}(\mathbf{x_c})-\mathcal{H}(\mathbf{x_c}|\mathbf{x})=\mathcal{H}(\mathbf{x_c})-\frac{1}{2}\log((2\pi e)^n|\Sigma|) \tag{2}$$

Where $\mathcal{H}(\mathbf{x_c}|\mathbf{x})$ is the entropy of the added Gaussian noise. Here $|\Sigma|$ denotes the determinant of the covariance matrix. Then by applying Theorem A.1 (from the appendix) which gives an upper bound for the entropy, to $\mathbf{x_c}$, we can write:

$$I(\mathbf{x_c};\mathbf{u})\leq\frac{1}{2}\log((2\pi e)^n\frac{|Cov(\mathbf{x_c})|}{|\Sigma|}) \tag{3}$$

Since $\mathbf{x}$ and $\mathbf{r}$ are independent variables and $\mathbf{x_c}=\mathbf{x}+\mathbf{r}$, we have $|Cov(\mathbf{x_c})|=|Cov(\mathbf{x})+\Sigma|$. In addition, since covariance matrices are positive semi-definite, we can get the eigen decomposition of $Cov(\mathbf{x})$ as $Q\Lambda Q^T$ where the diagonal matrix Λ has the eigenvalues. Since Σ is already a diagonal matrix, $|Cov(\mathbf{x})+\Sigma| = |Q(\Lambda+\boldsymbol{\sigma}^2)Q^T| = \prod_{i=1}^n(\lambda_i+\sigma_i^2)$. By substituting this in Equation 3, and simplifying we get the upper bound for $I(\mathbf{x_c};\mathbf{u})$ as the following:

$$I(\mathbf{x_c};\mathbf{u})\leq\frac{1}{2}\log((2\pi e)^n\prod_{i=1}^n(1+\frac{\lambda_i}{\sigma_i^2})) \tag{4}$$

3.2 Lower bound on $I(\mathbf{x_c};\mathbf{c})$

Theorem 3.1. *The lower bound on $I(\mathbf{x_c};\mathbf{c})$ is:*

$$\mathcal{H}(\mathbf{c})+\max_q \mathbb{E}_{\mathbf{x_c},\mathbf{c}}[\log q(\mathbf{c}|\mathbf{x_c})] \tag{5}$$

Where q denotes all members of a possible family of distributions for this conditional probability.

Proof. The lemma and accompanying proof for this theorem are in the appendix. □

3.3 Loss Function

Now that we have the upper and lower bounds, we can reduce our problem to the following optimization where we minimize the upper bound (Equation 4) and maximize the lower bound (Equation 5):

$$\min_{\boldsymbol{\sigma},q} \quad \frac{1}{2}\log((2\pi e)^n\prod_{i=1}^n(1+\frac{\lambda_i}{\sigma_i^2}))+\lambda\sum_{\mathbf{c_i},\mathbf{x_{c_i}}}(-\log q(\mathbf{c_i}|\mathbf{x_{c_i}})) \tag{6}$$

We omit the $\mathcal{H}(\mathbf{c})$ from the lower bound in Equation 5, since it is a constant. We also write the expected value in the same equation in the form of a summation over all possible representations and conducive features. To make this summation tractable, in our loss function we replace this part of the formulation with the empirical cross-entropy loss of the target classifier over all training examples. In other words, the loss of preserving the conducive features is substituted by the classification loss for those features. We also relax the optimization further by rewriting the first term. Since minimizing this term is equivalent to maximizing the standard deviation of the noise, we change the fraction into a subtraction. Our final loss function becomes:

$$\mathcal{L}=-\log\frac{1}{n}\sum_{i=0}^n\sigma_i^2+\lambda\mathbb{E}_{\mathbf{r}\sim\mathcal{N}(\boldsymbol{\mu},\sigma^2),\mathbf{x}\sim\mathcal{D}}\Big[-\sum_{k=1}^K y_k\log(f_\theta(\mathbf{x}+\mathbf{r}))_k\Big] \tag{7}$$

The second term is the expected cross-entropy loss, over the randomness of the noise and the data instances. The variable $\boldsymbol{\mu}$ is the mean of the noise distributions. The variable K is the number of classes for the target task, and y_k is the indicator variable that determines if a given example belongs to class k. More intuitively, the first term increases the noise of each feature and provides privacy. The second term decreases the classification error and maintains accuracy. The parameter λ is a knob that provides a trade-off between these two.

3.4 Suppressed Representation

After finding the noisy representation $\mathbf{x_c}$, we use it to generate the final suppressed representation $\mathbf{x_s}$. By applying a cutoff threshold T on $\boldsymbol{\sigma}$, we generate binary mask $\mathbf{b}$ such that $b_i=1$ if $\sigma_i\geq T$, and $b_i=0$ otherwise. We create representation $\mathbf{x_s}=(\mathbf{x}+\mathbf{r})\circ\mathbf{b}+\mu_\mathbf{s}$, where $\mathbf{r}\sim\mathcal{N}(0,\boldsymbol{\sigma})$ and $\mu_\mathbf{s}$ are constant values that are set to replace non-conducive features. According to the data processing inequality [7], the upper bound on $I(\mathbf{x_c};\mathbf{u})$ holds for $\mathbf{x_s}$ as well, since $I(\mathbf{x_s};\mathbf{u})\leq I(\mathbf{x_c};\mathbf{u})$. The same inequality also implies that the lower bound achieved for $I(\mathbf{x_c};\mathbf{c})$ does not necessarily hold for $\mathbf{x_s}$. To address this, we write another optimization problem, to find $\mu_\mathbf{s}$ such that cross entropy loss, i.e, $\min_{\mu_\mathbf{s}}\sum_{k=1}^K y_k\log(f_\theta(\mathbf{x_s}))_k$ is minimized. Solving this guarantees that the lower bound of Equation 5 also holds for $I(\mathbf{x_s};\mathbf{c})$.

4 CLOAK FRAMEWORK

This section describes Cloak's framework in more detail. Cloak comprises of two phases: first, an offline phase where we solve the optimization problems to find the conduciveness of the features and the suppression constant values. Second, an online prediction phase where the non-conducive features in a given input are suppressed and a sifted and a suppressed representation of the data is sent to the remote target service provider for prediction. In this section we

discuss details of these two phases, starting from the details of the offline phase.

4.1 Noise Re-parameterization and Constraints

To solve the optimization problem of Section 3, Cloak's approach is to cast the noise distribution parameters as trainable tensors, making it possible to solve the problem using conventional gradient-based methods. To be able to define gradients over the means and variances, we rewrite the noise sampling to be $\mathbf{r} = \boldsymbol{\sigma} \circ \mathbf{e} + \boldsymbol{\mu}$, instead of $\mathbf{r} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, where $\mathbf{e} \sim \mathcal{N}(0,1)$. The symbol $\circ$ denotes the element-wise multiplication of elements of $\boldsymbol{\sigma}$ and $\mathbf{e}$. This redefinition enables us to formulate the problem as an analytical function for which we can calculate the gradients. We also need to reparameterize $\boldsymbol{\sigma}$ to limit the range of standard deviation of each feature (σ). If it is learned through a gradient-based optimization, it can take on any value, while we know that variance can not be negative. In addition, we also do not want the σs to grow over a given maximum, M. We put this extra constraint on the distributions, to limit the σs from growing infinitely (to decrease the loss), taking the growth opportunity from the standard deviation of the other features. Finally, we define a trainable parameter $\boldsymbol{\rho}$ and write $\boldsymbol{\sigma} = \frac{1.0+\tanh(\boldsymbol{\rho})}{2} M$, where the tanh function is used to constraint the range of the σs, and the addition of 1 is to guarantee the positivity of the variance.

4.2 Cloak's Perturbation Training Workflow

Algorithm 1 shows the steps of Cloak's optimization process. This algorithm takes the training data ($\mathcal{D}$), labels (y), a pre-trained model (f_θ), and the privacy-utility knob (λ) as input, and computes the optimized tensor for noise distribution parameters. During the initialization step, the algorithm sets the trainable tensor for the means ($\boldsymbol{\mu}$) to 0 and initializes the substitute trainable tensor ($\boldsymbol{\rho}$) with a large negative number. This generates the initial value of zero for the standard deviations.

In each step of the optimization, the algorithm calculates the loss function on a batch of training data and computes the gradient of the loss with respect to the $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ by applying backpropagation. Since the loss (Equation 7) incorporates expected value over noise samples, Cloak uses Monte Carlo sampling [34] with a sufficiently large number of noise samples to calculate the loss. This means that to apply a single update to the trainable parameters, Cloak runs multiple forward passes on the entire classifier, at each pass draws new samples for the noise tensor (the elements of which are independently drawn), and averages over the losses and applies the update using the average. However, in practice, if mini-batch training is used, only a single noise sample for each update can yield desirable results, since a new noise tensor is sampled for each mini-batch. Once the training is finished, the optimized mean and standard deviation tensors are collected and passed to the next phase.

4.3 Feature Sifting and Suppression

For sifting the features we use the trained standard deviation tensor ($\boldsymbol{\sigma}$), which we call "noise map". A high value in the noise map for a feature indicates that the feature is less important. Different noise maps are created by changing the privacy-utility knob (λ). We use a cutoff threshold T, to map the continuous spectrum of values of a noise map, to binary values ($\mathbf{b}$). While choosing the cutoff threshold (T) depends on the privacy-utility trade-offs, in practice, finding the optimal value for T is not challenging. That is because the trained σs are easy to be sifted as they are pushed to either side of the spectrum, i.e., they either have a very large (near M) or a very small value (near 0). See Section 5.6 for more details.

To suppress the non-conducive features, one simple way is to send the noisy representations, i.e, adding noise from the $(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ to the input to get the x_c representations that are sent out for prediction. This method, however, suffers from two shortcomings: first, it does not directly suppress and remove the features, which could leave the possibility of data leakage. Second, because of the high standard deviations of noise, in some cases, the generated representation might be out of the domain of the target classifier, which could have negative effects on the utility. Another way of suppressing the non-conducive features is to replace them with zeros (black pixels in images for example). This scheme also, suffers from potential accuracy degradation, as the values we are using for suppression (i.e. the zeros) might not match the distribution of the data that the classifier expects.

To address this, we find a suppressed representation (Section 3.4), i.e., we train the constant suppression values that need to replace the non-conducive features. Intuitively, these learned values reveal what the target classifier perceives as common among all the inputs from the training set, and what it expects to see. Algorithm 2 shows the steps of this training process. The algorithm finds $\boldsymbol{\mu}_s$, the values by which we replace the non-conducive features. The only objective of this training process is to increase the accuracy, therefore we use the cross-entropy loss as our loss function.

Algorithm 1 Perturbation Training

Input: $\mathcal{D}$, y, f_θ, m, λ
Initialize $\boldsymbol{\mu} = 0$, $\boldsymbol{\rho} = -10$ and $M \geq 0$
repeat
  Select training batch $\mathbf{x}$ from $\mathcal{D}$
  Sample $\mathbf{e} \sim \mathcal{N}(0,1)$
  Let $\boldsymbol{\sigma} = \frac{1.0+\tanh(\boldsymbol{\rho})}{2}(M)$
  Let $\mathbf{r} = \boldsymbol{\sigma} \circ \mathbf{e} + \boldsymbol{\mu}$
  Take gradient step on $\boldsymbol{\mu}$, $\boldsymbol{\rho}$ from Eq. (7)
until Algorithm converges
Return: $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$

Algorithm 2 Suppression-Value Training

Input: $\mathcal{D}$, y, f_θ, $\boldsymbol{\sigma}$, $\boldsymbol{\mu}$, $\mathbf{b}$
Initialize $\boldsymbol{\mu_s} = \boldsymbol{\mu}$
repeat
  Select training batch $\mathbf{x}$ from $\mathcal{D}$
  Sample $\mathbf{r} \sim \mathcal{N}(0, \boldsymbol{\sigma}^2)$
  Let $\mathbf{x_s} = (\mathbf{x}+\mathbf{r}) \circ b + \boldsymbol{\mu_s}$
  Take gradient step on $\boldsymbol{\mu}_s$ from $\mathbb{E}_r[\mathcal{L}_{CE}(f_\theta(\mathbf{x_s}), y)]$
until Algorithm converges
Return: $\boldsymbol{\mu}_s$

4.4 Online Prediction

The prediction (inference) phase is when unseen test inputs that we protect are sent to the remote service provider for classification. This process is computationally efficient; it only adds noise sampling, masking, and addition to the normal conventional prediction process. First, a noise tensor sampled from the optimized distribution $\mathcal{N}(0,\sigma^2)$ is added to the input, then the binary mask b is applied to the noisy input image. Finally, μ_s is added to $\mathbf{x}$ and the resulting sifted representation is sent to the service provider. As an example, the last row of Figure 1 shows representations generated by Cloak, for different tasks, using the noise maps from the third row. As the images show, the non-conducive features are removed and replaced with μ_s. The conducive features, however, are visible.

5 EXPERIMENTAL RESULTS

To evaluate Cloak, we use four real-world datasets on four Deep Neural Networks (DNNs). Namely, we use VGG-16 [76] and ResNet-18 [26] on CelebA [47], AlexNet [38] on CIFAR-100 [37], a modified version of VGG-16 model on UTKFace [87], and LeNet-5 [39] on MNIST [40]. The mutual information numbers reported in this section are estimated over the test set using the Shannon Mutual Information estimator provided by the Python ITE toolbox [79]. For the experiments that are devised to compare Cloak with previous work, Shredder [52], in order to create a similar setup, we apply Cloak to the last convolution layer of the DNN and create *sifted intermediate representations* which are then sent to the target classifier. In the other experiments, Cloak is applied directly to the input images. Code and information about hyper-parameters used in each of the experiments is provided in the appendix.

5.1 Detailed Experimental Setup

In this section, we elaborate on the details of our experimental setup. This includes dataset specifications, hardware and OS specifications, neural network architectures, and finally, mutual information estimation.

5.1.1 Dataset Specifications. There are four datasets used in our evaluations: CelebA [47], CIFAR-100 [37], UTKFace [87] and MNIST [40]. We have used these datasets with VGG-16 [76], ResNet-18 [26], AlexNet [38], VGG-16 (modified), and LeNet-5 [39] neural networks, respectively. We define a set of target prediction tasks over these datasets. Specifically, we use smile detection, black-hair color classification, and eyeglass detection on CelebA, the 20 super-class classification on CIFAR-100, and gender detection on UTKFace. For MNIST, we use a classifier that detects if the input is greater than five and another one that classifies what the input digit actually is. The accuracy numbers reported in this section are all on a held-out test set, which has not been seen during training by the neural networks. For Cloak results, since the output is not deterministic, we repeatedly run the prediction ten times on the test set with the batch size of one and report the mean accuracy. Since the standard deviation of the accuracy numbers is small (consistently less than 1.0%) the confidence bars are not visible on the graphs. The input image sizes for CelebA, CIFAR-100, UTKFace and MNIST are 224×224×3, 32×32×3, 32×32×3, and 32×32, respectively. In addition, in our experiments, the inputs are all normalized to 1. The experiments are all carried out using Python 3.6 and PyTorch 1.3.1. We use Adam optimizer for perturbation training.

5.1.2 Experimentation Hardware and OS. We have run the experiments for CelebA dataset on an Nvidia RTX 2080 Ti GPU, with 11GB VRAM, paired with 10 Intel Core i9-9820X processors with 64GBs of memory. The rest of the experiments were run on the CPU. The system runs an Ubuntu 18.04 OS, with CUDA version V10.2.89.

5.1.3 Neural Network Architectures. The code for all the models is available in the supplementary materials. The VGG-16 for UTKFace is different from the conventional one in the size of the last 3 fully connected layers. They are (512,256), (256,256) and (256,2). The pre-trained accuracy of the networks for smile detection, super-class classification, gender detection, and greater than five detection are 91.8%, 55.7%, 87.87%, and 99.29%.

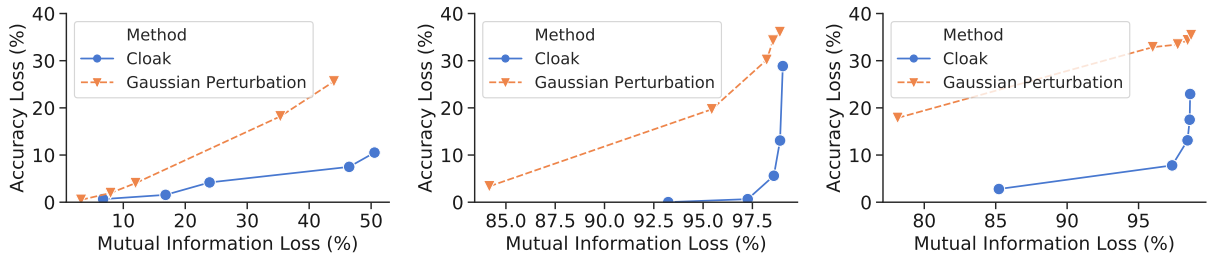
5.1.4 Mutual Information Estimation. The mutual information between the input images and their noisy representations are estimated over the test set images using ITE [79] toolbox's Shannon mutual information estimator. For MNIST images, our dataset has inputs of size 32×32 pixels, which we flatten to 1024 element vectors, for estimating the mutual information. For other datasets, since the images are larger (32×32×3), there are more dimensions and mutual information estimation is not accurate. So, we calculate mutual information channel by channel (i.e. we estimate the mutual information between the red channel of the image and its noisy representation, then the green channel and then blue), and we average over all channels.

The numbers reported in 5.2 are mutual information loss percentages, which means the lost mutual information among the publicized image and the original one is divided by the information content in the original images. This information content was estimated using self-information (Shannon information), using the same toolbox.

5.2 Privacy-Accuracy Trade-Off

Figure 2 shows accuracy loss of the DNN classifiers using sifted representations vs. the loss in mutual information. This is the loss in mutual information between the original image and its noisy representation, divided by the amount of information in bits in the original image. The target tasks are 20 superclass classification for CIFAR-100, > 5 classification for MNIST and gender classification for UTKFace. In this experiment, we compare Cloak to adding Gaussian perturbation of mean zero and different standard deviations to all pixels of the images. For fair comparison, we choose Cloak's suppression with noisy representations. For MNIST and UTKFace, Cloak reduces the information in the input significantly (93% and 85% respectively) with little loss in accuracy (0.5% and 2.7%). In CIFAR-100, the accuracy is slightly more sensitive to the mutual information loss. This is due to the difference in the classification tasks. The tasks for MNIST and UTKFace have only two classes, while for CIFAR-100, the classifier needs to distinguish between 20 classes.

For all three datasets, we see that Cloak achieves a significantly higher accuracy for same loss in mutual information compared to Gaussian perturbation. This is because Cloak adds more noise to the irrelevant features, and less to the relevant ones, whereas Gaussian perturbations are added uniformly across the input. We do not present mutual information results for the CelebA dataset here, since the input images have an extremely large number of features and



(a) CIFAR-100 (b) MNIST (c) UTKFace

Figure 2: Privacy-accuracy trade-off for CIFAR-100, MNIST and UTKFace dataset.

the mutual information estimator tool is not capable of estimating the mutual information accurately.

5.3 Adversary to Infer Information

To further evaluate the effectiveness of the representations that Cloak generates, we devise an experiment in which an adversary tries to infer properties of the sifted representations using a DNN classifier. We assume two adversary models here. First, the adversary has access to a unlimited number of samples from the sifted representations, therefore she can re-train her classifier to regain accuracy on the sifted representations. Second, a model in which the adversary's access to the sifted representation is limited and therefore she cannot retrain her classifier on the sifted representations. In this experiment, we choose smile detection as the target prediction task for which Cloak generates representations. Then, we model adversaries who try to discover two properties from the sifted representations: whether people in images wear glasses or not and whether their hair is black or not. The adversaries have pre-trained classifiers for both these tasks. The classifiers are VGG-16 DNNs, with accuracy of 96.4% and 88.2% for glasses and hair color classification, respectively.

Figure 3 shows the results of this experiment. Each point in this figure is generated using a noise map with a Suppression Ration (SR) noted in the figure. Higher SR means more features are suppressed. When adversaries do not retrain their models, using sifted representations with 95.6% suppression ratio causes the adversaries to almost completely lose their ability to infer eyeglasses or hair color and reach to the random classifier accuracy (50%). This is achieved while the target smile detection task only loses 5.16% accuracy. When adversaries retrain their models, using representations with slightly higher suppression ratio (98.3%) achieves the same goal. But this time, the accuracy of the target task drops to 78.9%. With the same suppression ratio, the adversary who tries to infer hair color loses more accuracy than the adversary who tries to infer eyeglasses. This is because, as shown in Figure 1, the conducive features of smile overlap less with the conducive features of hair than with the conducive features of eyeglasses.

5.4 Black-Box Access Mode

To show the applicability of Cloak, we show that it is possible for Cloak to protect users' privacy even when we have limited access to the target model. We consider a black-box setting in which we assume Cloak does not have any knowledge of the target model architecture

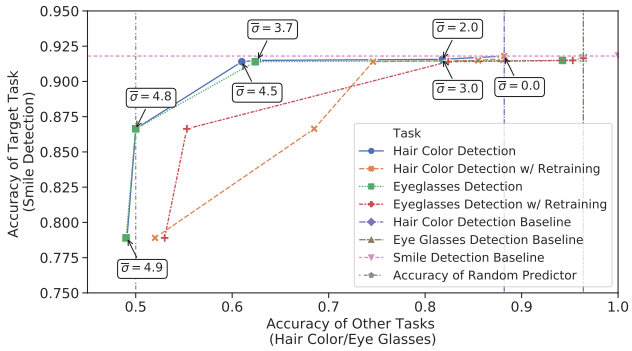


Figure 3: Cloak's protection for target task of smile detection (CelebA dataset) against adversaries that try to infer black-hair color or wearing of eyeglasses from the sifted representations.

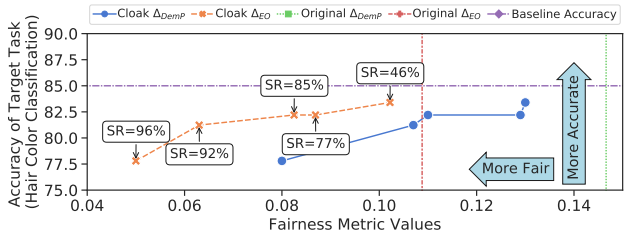


Figure 4: Effects of Cloak on fairness

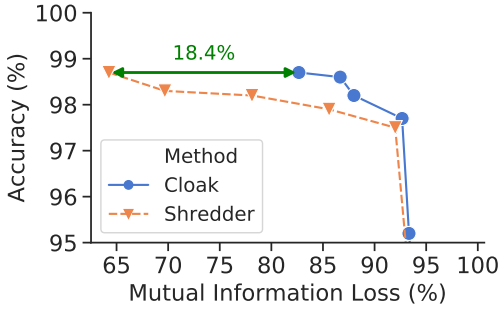


Figure 5: Comparison to Shredder [52]

or its parameters and is only allowed to send requests and get back responses. In this setting, we first train a substitute model that helps us to train Cloak's representations. Note that training a substitute model for black-box setting is a well-established practice in the context of adversarial examples [49, 63] and inference attacks [28, 75]. The main challenge is generating the training data needed for training the substitute model. However, that has been already addressed in previous work and we follow a similar methodology to the methodology described in Shokri et al. [75]. We divide the original dataset (CelebA) into two equal-size disjoint training sets, one for the target and the other for the substitute model. We assume a target service provider that has two ResNet18 [26] DNNs deployed, one for the task of black hair color classification, and one for smile detection. Since we assume no knowledge of the model architecture, Cloak substitutes the target classifiers with another architecture, i.e, with two VGG-16 DNNs. Cloak substitute models for the hair and smile tasks have accuracies of 84.9% and 90.9% and the target models have accuracies of 87.3% and 91.8%. After training the substitute model, we apply Cloak to them to find noise maps and suppressed representations.

Figure 6c and 6d show the results for these experiments. Cloak performs similarly effective in both white-box and black-box settings and for both hair color classification and smile detection tasks. The reason is that the DNN classifiers of the same task are known to learn similar patterns and decision boundaries [3, 63]. For the smile detection, we can see that with suppression ratio of 33%, The Cloak black-box generated representations can get prediction accuracy of 91.3%, even higher than the baseline prediction accuracy of the classifier it is produced from. That is because the generated representations are fed to the target classifier, which has a higher baseline accuracy than the substitute model.

5.5 Post-hoc Effects of Cloak on Fairness

Cloak, by removing extra features, not only benefits privacy but can also remove unintended biases of the classifier, resulting in a more fair classification. In many cases the features that bias the classifiers highly overlap with the non-conducive features that Cloak discovers. Therefore, applying Cloak can result in predictions that are more fair, without the need to change the classifier. This subsection evaluates this positive side-effect of Cloak by adopting a setup similar to that of Kairouz et al. [31]. We measure the fairness of the black-hair color classifier using the sifted representations, while considering gender to be a sensitive variable that can cause bias. We use two metrics for our experiments, the difference in Demographic Parity (Δ_{DemP}), and the difference in Equal Opportunity (Δ_{EO}). More details on the metrics and the measurements can be found in the supplementary material. Figure 4 shows that as Cloak suppresses more non-conducive features, the fairness metrics improve significantly. We see 0.05 reduction in both metrics due to the removal of gender related non-conducive features. It is noteworthy that the biasing features in the hair color classifier are not necessarily the gender features shown in Figure 1. Those features show what a gender classifier uses to make its decision.

5.6 Thresholds, Suppression Mechanisms, and Comparison to Shredder

Sensitivity to threshold values. Figure 6a shows the effect of different thresholds (T) values on suppression ratio of features on smile detection (on CelebA/ VGG-16). Different series show different noise maps attained with different values of λ. $\bar{\sigma}$ denotes the average standard deviation of a noise map, and the parameter M (maximum standard deviation) of Section 4.1 is set to 5. The figure shows that the choice of T is not critical and in fact is a simple task, since it has little effect on the subset of features that get suppressed. This is because during the training of perturbation parameters, the standard deviations are pushed to the either sides of the spectrum (close to 0 or close to M).

Different suppression schemes. Figure 6b shows the accuracy of three suppression schemes described in Section 4.3 on the smile detection task (on CelebA/ VGG-16). Among different schemes, suppression using the trained values yields better accuracy for the same suppression ratio, since it captures what the classifier expects to receives. Suppression with noise (sending noisy representations) performs slightly worse than training, and that is mainly due to the uncertainty brought by the noise.

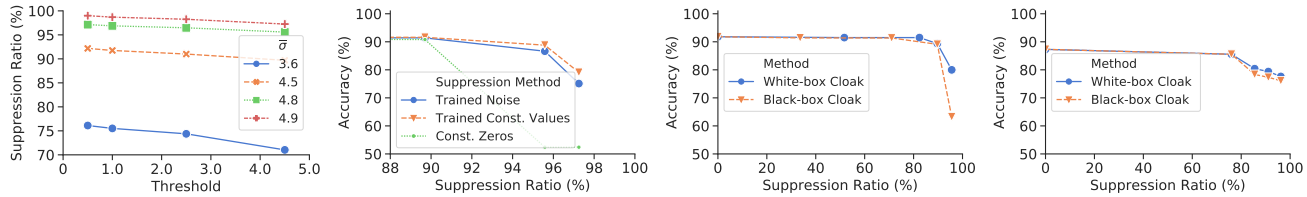
Comparision to Shredder. Figure 5 compares Cloak and Shredder [52] on the MNIST dataset using LeNet for the target task of digit classification. To create a fair setup, we deploy Cloak to the output of the last convolution layer of LeNet, similar to Shredder. Cloak achieves a significantly higher accuracy for same levels of MI loss, which shows the effectiveness of Cloak, in the intermediate representation space. For the initial point where there is almost no loss in accuracy, Cloak achieves 18.4% more information loss. This better performance is partly due to directly learning the importance of each feature, as opposed to generating patterns similar to a collection that yields high accuracy. It is also partly due to the extra step that Cloak takes at learning the constant suppression values, which ensures the generated representations are in the domain of the classifier.

6 RELATED WORK

This section reviews related work on the privacy of web services. The section first briefly discusses the privacy of web applications in general, and then more thoroughly discusses privacy in the context of machine learning.

6.1 Web-application Privacy

Despite the privacy issues, sharing personal content on the web unfortunately is still common. Therefore, researchers lavished attention on the research that makes such sharing safe, secure, and private [4, 17]. Mannan et al. [51] proposed a method that focuses on privacy-enhanced web content sharing in any user-chosen web server. There is also a body of work that conducts longitudinal studies on deleted web content and their subsequent information leakage [6, 57]. The research in this area focuses on data leakage through social media [73, 88], blogging services that publish information [83], or aggregation of web data [66]. Cloak, however, focuses on an inference-as-a-service setup where private queries that potentially contain sensitive information are sent to a web-service to run machine learning inference.



(a) Threshold Sensitivity **(b) Suppression Schemes** **(c) Black-box-smile detection** **(d) Black-box-hair color**

Figure 6: (a) shows the effect of different thresholds on suppression rate. (b) compares different suppression methods. (c) and (d) show performance of Cloak in a black-box setting.

6.2 Machine Learning Privacy

Privacy-preserving machine learning research can be broadly categorized based on the phase on which they focus, i.e., training vs prediction. The majority of these studies fall under the training category [53] where they try to protect contributors' private data from getting embedded in the trained ML model [1, 13, 14, 29, 62, 74, 78] or from being published in public datasets [19–21]. However, the impending importance of prediction (inference) privacy has led to the emergence of recent research efforts in this direction [18, 27, 41, 60, 61, 86]. There is also a smaller body of work focused on the privacy of model architecture and parameters [11, 36], which is out of the scope of this paper. Below, the more related works are discussed in more detail.

Training phase. For *training*, the literature abounds with studies that use noise addition as a randomization mechanism to protect privacy [1, 13, 14, 21, 62, 64, 74]. Most notably, differential privacy [20], a mathematical framework that quantifies privacy, has spawned a vast body of research in noise-adding mechanisms. For instance, it has been applied to many machine learning algorithms, such as logistic regression [12], statistical risk minimization [13], principal component analysis [14, 29], and deep learning [1, 62, 64, 72, 74], to name a few. Many of these studies have applied differential privacy to a training setting where they are concerned with leaking private information in training set through the machine learning model. There is also a body of work focused on secure training of machine learning models using cryptographic protocols [2, 2, 25, 56, 70, 71].

Finally, there are also several privacy-enhancing mechanisms, such as Federated learning [32, 46] and Split learning [65, 77], which use gradients or abstract representations of data in lieu of raw inputs, to train ML models and enhance privacy. These methods have been coupled with differential privacy [5, 10, 67] or information-theoretic notions [84] to provide meaningful privacy guarantees.

Prediction/Inference privacy. Only a handful of studies have addressed privacy of prediction by adding noise to the data. Osia et al. [60] employed dimensionality reduction techniques to reduce the amount of information before sending it to an untrusted cloud service. Wang et al. [86] propose a noise injection framework that randomly nullifies input elements for private inference, but their method requires retraining of the entire network. Leroux et al. [41] propose an autoencoder to randomize the data, but the intensity of their obfuscation is too small to be irreversible, as they state.

Liu et al. [44] propose DEEProtect, an information-theoretic method which offers two usage modes for protecting privacy. One where it assumes no access to the privacy-sensitive inference labels and one where it assumes access to the privacy-sensitive labels. Deeprotect incorporates the sensitive inference into its formulation for the latter usage mode. A more recent work, Shredder [52], proposes to *heuristically* sample and reorder additive noise at run time based on the previously collected additive tensors that the DNN can tolerate (anti-adversarial patterns). In contrast, Cloak's approach is to directly reduce information by learning conducive features and suppressing non-conducive ones with learned constant values. We also experimentally show that Cloak outperforms this prior work. More importantly, this prior work relies on executing parts of the network on the edge side and sending the results to the cloud. However, this separation is not always possible, as the service providers might not be willing to share the model parameters or change their infrastructure to accommodate for this method. Also, in some cases, the edge device might be incapable of running the first convolution layers of the neural network. In contrast, we show that Cloak can perform equally efficiently in black-box settings without the collaboration of the service provider.

Privacy on offloaded computation can also be provided by the means of cryptographic tools such as homomorphic encryption and/or Secure Multiparty Computation (SMC) [9, 18, 23, 30, 45, 48, 54, 85]. However, these approaches suffer from a prohibitive computational cost (Table 1), on both the cloud and user side, exacerbating the complexity and compute-intensity of neural networks especially on resource-constrained edge devices. Cloak, in contrast, avoids the significant cost of encryption and homomorphic data processing.

Several other research [24, 58, 82] rely on trusted execution environments to remotely run ML algorithms. However, this model requires the users to send their data to an enclave running on remote servers and is vulnerable to the new breaches in hardware [35, 43].

7 CONCLUSION

The surge in the use of machine learning is driven by the growth in data and compute power. The data mostly comes from people [81] and includes an abundance of private information. We propose Cloak, a mechanism that finds features in the data that are unimportant and non-conducive for a cloud ML prediction model. This enables Cloak to suppress those features before sending them to the cloud, providing only the minimum information exposure necessary to receive the particular service. In doing so, Cloak not only minimizes the impact on the utility of the service, but it also imposes minimal overhead on the response time of the prediction service.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful suggestions and comments. This work was in part supported by National Science Foundation (NSF) awards CNS#1703812, ECCS#1609823, CCF#1553192, CCF#1823444, Air Force Office of Scientific Research Young Investigator Program (YIP) award #FA9550-17-1-0274, National Institute of Health (NIH) award #R01EB028350, and Air Force Research Laboratory (AFRL) and Defense Advanced Research Project Agency (DARPA) under agreement number #FA8650-20-2-7009 and #HR0011-18-C-0020. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Samsung, Amazon, NSF, AFSOR, NIH, AFRL, DARPA, or the U.S. Government.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *ACM Conference on Computer and Communications Security (CCS)*.

[2] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J Kusner, and Adrià Gascón. 2019. QUOTIENT: two-party secure neural network training and prediction. In *ACM Conference on Computer and Communications Security (CCS)*.

[3] Devansh Arpit, Stanislaw Jastrzkebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. 2017. A closer look at memorization in deep networks. In *International Conference on Machine Learning (ICML)*.

[4] Babak Amin Azad, Pierre Laperdrix, and Nick Nikiforakis. 2019. Less is more: quantifying the security benefits of debloating web applications. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1697–1714.

[5] Borja Balle, Peter Kairouz, H Brendan McMahan, Om Thakkar, and Abhradeep Thakurta. 2020. Privacy amplification via random check-ins. *arXiv preprint arXiv:2007.06605* (2020).

[6] Timothy Barron, Najmeh Miramirkhani, and Nick Nikiforakis. 2019. Now You See It, Now You Don't: A Large-scale Analysis of Early Domain Deletions. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. USENIX Association, Chaoyang District, Beijing, 383–397. https://www.usenix.org/conference/raid2019/presentation/barron

[7] Normand J. Beaudry and Renato Renner. 2011. An intuitive proof of the data processing inequality. *arXiv preprint arXiv:1107.0740* (2011). arXiv:1107.0740 [quant-ph]

[8] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424* (2015).

[9] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. 2020. MP2ML: A Mixed-Protocol Machine Learning Framework for Private Inference. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (Virtual Event, Ireland) *(ARES '20)*. Association for Computing Machinery, New York, NY, USA, Article 14, 10 pages. https://doi.org/10.1145/3407023.3407045

[10] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[11] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. 2020. Cryptanalytic extraction of neural network models. In *Annual International Cryptology Conference*. Springer, 189–218.

[12] Kamalika Chaudhuri and Claire Monteleoni. 2009. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (Eds.). Curran Associates, Inc., 289–296. http://papers.nips.cc/paper/3486-privacy-preserving-logistic-regression.pdf

[13] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. 2009. Differentially Private Empirical Risk Minimization. *arXiv preprint arXiv:0912.0071* (2009). arXiv:0912.0071 [cs.LG]

[14] Kamalika Chaudhuri, Anand D. Sarwate, and Kaushik Sinha. 2013. A Near-Optimal Algorithm for Differentially-Private Principal Components. *J. Mach. Learn. Res.* 14, 1 (Jan. 2013), 2905–2943.

[15] Thomas M Cover and Joy A Thomas. 2012. *Elements of information theory*. John Wiley & Sons.

[16] Paul W. Cuff and Lanqing Yu. 2016. Differential Privacy as a Mutual Information Constraint. In *ACM Conference on Computer and Communications Security (CCS)*.

[17] Wei Dong, Minghui Qiu, and Feida Zhu. 2014. Who am I on twitter? a cross-country comparison. In *Proceedings of the 23rd International Conference on World Wide Web*. 253–254.

[18] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In *International Conference on Machine Learning (ICML)*.

[19] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy via Distributed Noise Generation. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques* (St. Petersburg, Russia) *(EUROCRYPT'06)*. Springer-Verlag, Berlin, Heidelberg, 486–503. https://doi.org/10.1007/11761679_29

[20] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Third Conference on Theory of Cryptography* (New York, NY) *(TCC'06)*. Springer-Verlag, Berlin, Heidelberg, 265–284. https://doi.org/10.1007/11681878_14

[21] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9 (Aug. 2014), 211–407. https://doi.org/10.1561/0400000042

[22] Facebook. 2019. A research tool for secure machine learning in PyTorch. online–accessed June 2020, url: https://crypten.ai.

[23] Bo Feng, Qian Lou, Lei Jiang, and Geoffrey C Fox. 2020. CryptoGRU: Low Latency Privacy-Preserving Text Analysis With GRU. *arXiv preprint arXiv:2010.11796* (2020).

[24] Lucjan Hanzlik, Yang Zhang, Kathrin Grosse, Ahmed Salem, Max Augustin, Michael Backes, and Mario Fritz. 2018. MLCapsule: Guarded Offline Deployment of Machine Learning as a Service. *arXiv preprint arXiv:1808.00590* (2018). arXiv:1808.00590 [cs.CR]

[25] Hanieh Hashemi, Yongqin Wang, and Murali Annavaram. 2020. DarKnight: A Data Privacy Scheme for Training and Inference of Deep Neural Networks. *arXiv preprint arXiv:2006.01300* (2020).

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[27] Zecheng He, Tianwei Zhang, and Ruby B Lee. 2019. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*. 148–162.

[28] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. 2019. MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples. In *ACM Conference on Computer and Communications Security (CCS)*.

[29] Xiaoqian Jiang, Zhanglong Ji, Shuang Wang, Noman Mohammed, Samuel Cheng, and Lucila Ohno-Machado. 2013. Differential-private data publishing through component analysis. *Transactions on data privacy* 6, 1 (2013), 19.

[30] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In *USENIX Security Symposium (USENIX Security)*.

[31] Peter Kairouz, Jiachun Liao, Chong Huang, and Lalitha Sankar. 2019. Censored and Fair Universal Representations using Generative Adversarial Models. *arXiv preprint arXiv:1910.00411* (2019). arXiv:1910.00411 [cs.LG]

[32] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

[33] K. Kalantari, L. Sankar, and O. Kosut. 2017. On information-theoretic privacy with general distortion cost functions. In *2017 IEEE International Symposium on Information Theory (ISIT)*. 2865–2869. https://doi.org/10.1109/ISIT.2017.8007053

[34] Malvin H. Kalos and Paula A. Whitlock. 1986. *Monte Carlo Methods. Vol. 1: Basics*. Wiley-Interscience, USA.

[35] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2019. Spectre Attacks: Exploiting Speculative Execution. In *IEEE Symposium on Security and Privacy (S&P)*.

[36] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. 2019. Thieves on sesame street! model extraction of bert-based apis. *arXiv preprint arXiv:1910.12366* (2019).

[37] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. [n.d.]. CIFAR-100 (Canadian Institute for Advanced Research). ([n. d.]). http://www.cs.toronto.edu/~kriz/cifar.html url: http://www.cs.toronto.edu/~kriz/cifar.html.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60 (2012), 84–90.

[39] Yann LeCun. 1998. Gradient-based learning applied to document recognition.

[40] Yann LeCun and Corinna Cortes. [n.d.]. The MNIST Dataset Of Handwritten Digits. online accessed May 2019 http://www.pymvpa.org/datadb/mnist.html.

[41] Sam Leroux, Tim Verbelen, Pieter Simoens, and Bart Dhoedt. 2018. Privacy Aware Offloading of Deep Neural Networks. arXiv:1805.12024 [cs.LG]

[42] Jiachun Liao, Oliver Kosut, Lalitha Sankar, and Flávio P. Calmon. 2017. A General Framework for Information Leakage.

[43] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. 2018. Meltdown: Reading Kernel Memory from User Space. In *USENIX Security Symposium (USENIX Security)*.

[44] C. Liu, S. Chakraborty, and P. Mittal. 2017. DEEProtect: Enabling Inference-based Access Control on Mobile Sensing Applications. *ArXiv* abs/1702.06159 (2017).

[45] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. 2017. Oblivious neural network predictions via minionn transformations. In *ACM Conference on Computer and Communications Security (CCS)*.

[46] Zaoxing Liu, Tian Li, Virginia Smith, and Vyas Sekar. 2019. Enhancing the privacy of federated learning with sketching. *arXiv preprint arXiv:1911.01812* (2019).

[47] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *International Conference on Computer Vision (ICCV)*.

[48] Qian Lou, Bian Song, and Lei Jiang. 2020. AutoPrivacy: Automated Layer-wise Parameter Selection for Secure Neural Network Inference. *arXiv preprint arXiv:2006.04219* (2020).

[49] Jiajun Lu, Theerasit Issaranon, and David Forsyth. 2017. Safetynet: Detecting and rejecting adversarial examples robustly. In *International Conference on Computer Vision (ICCV)*.

[50] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. 2018. Learning adversarially fair and transferable representations. *arXiv preprint arXiv:1802.06309* (2018).

[51] Mohammad Mannan and Paul C. van Oorschot. 2008. Privacy-Enhanced Sharing of Personal Content on the Web. In *Proceedings of the 17th International Conference on World Wide Web* (Beijing, China) *(WWW '08)*. Association for Computing Machinery, New York, NY, USA, 487–496. https://doi.org/10.1145/1367497.1367564

[52] Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Ali Jalali, Dean Tullsen, and Hadi Esmaeilzadeh. 2020. Shredder: Learning Noise Distributions to Protect Inference Privacy. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[53] Fatemehsadat Mireshghallah, Mohammadkazem Taram, Praneeth Vepakomma, Abhishek Singh, Ramesh Raskar, and Hadi Esmaeilzadeh. 2020. Privacy in Deep Learning: A Survey. In *ArXiv*, Vol. abs/2004.12254.

[54] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A Cryptographic Inference Service for Neural Networks. In *USENIX Security Symposium (USENIX Security)*. https://www.usenix.org/conference/usenixsecurity20/presentation/mishra

[55] MLPerf Organization. 2020. MLPerf Benchmark Suite. url: https://mlperf.org.

[56] P. Mohassel and Y. Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *IEEE Symposium on Security and Privacy (S&P)*.

[57] Mainack Mondal, Johnnatan Messias, Saptarshi Ghosh, Krishna P Gummadi, and Aniket Kate. 2016. Forgetting in social media: Understanding and controlling longitudinal exposure of socially shared data. In *Twelfth Symposium on Usable Privacy and Security ({SOUPS} 2016)*. 287–299.

[58] Krishna Giri Narra, Zhifeng Lin, Yongqin Wang, Keshav Balasubramaniam, and Murali Annavaram. 2019. Privacy-Preserving Inference in Machine Learning Services Using Trusted Execution Environments. *arXiv preprint arXiv:1912.03485* (2019).

[59] Alyssa Newcomb. 2018. Facebook data harvesting scandal widens to 87 million people. online–accessed February 2020, url:https://www.nbcnews.com/tech/tech-news/facebook-data-harvesting-scandal-widens-87-million-people-n862771.

[60] S. A. Osia, A. S. Shamsabadi, S. Sajadmanesh, A. Taheri, K. Katevas, H. R. Rabiee, N. D. Lane, and H. Haddadi. 2020. A Hybrid Deep Learning Architecture for Privacy-Preserving Mobile Analytics. *IEEE Internet of Things Journal* (2020), 1–1. https://doi.org/10.1109/JIOT.2020.2967734

[61] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Hamid R. Rabiee. 2020. Deep Private-Feature Extraction. *IEEE Transactions on Knowledge and Data Engineering* 32, 1 (Jan 2020), 54–66. https://doi.org/10.1109/tkde.2018.2878698

[62] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2016. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. *arXiv preprint arXiv:1610.05755* (2016). arXiv:1610.05755 [stat.ML]

[63] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *ACM on Asia conference on computer and communications security (AsiaCCS)*.

[64] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. *arXiv preprint arXiv:1802.08908* (2018).

[65] Maarten G. Poirot, Praneeth Vepakomma, K. Chang, J. Kalpathy-Cramer, R. Gupta, and R. Raskar. 2019. Split Learning for collaborative deep learning in healthcare. *ArXiv* abs/1912.12115 (2019).

[66] Vincent Primault, Vasileios Lampos, Ingemar Cox, and Emiliano De Cristofaro. 2019. Privacy-Preserving Crowd-Sourcing of Web Searches with Private Data Donor. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 1487–1497. https://doi.org/10.1145/3308558.3313474

[67] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Françoise Beaufays. 2020. Training production language models without memorizing user data. *arXiv preprint arXiv:2009.10031* (2020).

[68] Omer Rana and Joe Weinman. 2015. Data as a Currency and Cloud-Based Data Lockers. *IEEE Cloud Computing* 2, 2 (2015), 16–20.

[69] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. 2020. MLPerf Inference Benchmark. In *International Symposium on Computer Architecture (ISCA)*.

[70] Théo Ryffel, David Pointcheval, and Francis Bach. 2020. Ariann: Low-interaction privacy-preserving deep learning via function secret sharing. *arXiv preprint arXiv:2006.04593* (2020).

[71] Théo Ryffel, David Pointcheval, Francis Bach, Edouard Dufour-Sans, and Romain Gay. 2019. Partially encrypted deep learning using functional encryption. *Advances in Neural Information Processing Systems* 32 (2019), 4517–4528.

[72] Sina Sajadmanesh and Daniel Gatica-Perez. 2020. When Differential Privacy Meets Graph Neural Networks. *arXiv preprint arXiv:2006.05535* (2020).

[73] Sina Sajadmanesh, Sina Jafarzadeh, Seyed Ali Ossia, Hamid R Rabiee, Hamed Haddadi, Yelena Mejova, Mirco Musolesi, Emiliano De Cristofaro, and Gianluca Stringhini. 2017. Kissing cuisines: Exploring worldwide culinary habits on the web. In *Proceedings of the 26th international conference on world wide web companion*. 1013–1021.

[74] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *ACM Conference on Computer and Communications Security (CCS)*.

[75] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*.

[76] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).

[77] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and R. Raskar. 2019. Detailed comparison of communication efficiency of split learning and federated learning. *ArXiv* abs/1909.09145 (2019).

[78] Yeon sup Lim, M. Srivatsa, S. Chakraborty, and I. Taylor. 2018. Learning Light-Weight Edge-Deployable Privacy Models. *2018 IEEE International Conference on Big Data (Big Data)* (2018), 1290–1295.

[79] Zoltán Szabó. 2014. Information Theoretical Estimators Toolbox. *Journal of Machine Learning Research* 15 (2014), 283–287.

[80] M. Taram, A. Venkat, and D. Tullsen. 2020. Packet Chasing: Spying on Network Packets over a Cache Side-Channel. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 721–734. https://doi.org/10.1109/ISCA45697.2020.00065

[81] Stuart A. Thompson and Charlie Warzel. 2019. The Privacy Project: Twelve Million Phones, One Dataset, Zero Privacy. online–accessed February 2020, url: https://www.nytimes.com/interactive/2019/12/19/opinion/location-tracking-cell-phone.html.

[82] Florian Tramer and Dan Boneh. 2019. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=rJVorjCcKQ

[83] Tom Van Goethem, Najmeh Miramirkhani, Wouter Joosen, and Nick Nikiforakis. 2019. Purchased Fame: Exploring the Ecosystem of Private Blog Networks. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security* (Auckland, New Zealand) *(Asia CCS '19)*. Association for Computing Machinery, New York, NY, USA, 366–378. https://doi.org/10.1145/3321705.3329830

[84] Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. 2020. NoPeek: Information leakage reduction to share activations in distributed deep learning. *ArXiv* abs/2008.09161 (2020).

[85] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2020. FALCON: Honest-Majority Maliciously Secure Framework for Private Deep Learning. *arXiv preprint arXiv:2004.02229* (2020).

[86] Ji Wang, Jianguo Zhang, Weidong Bao, Xiaomin Zhu, Bokai Cao, and Philip S. Yu. 2018. Not Just Privacy. *ACM International Conference on Knowledge Discovery and Data Mining (KDD)* (2018). https://doi.org/10.1145/3219819.3220106

[87] Zhifei Zhang, Yang Song, and Hairong Qi. 2017. Age Progression/Regression by Conditional Adversarial Autoencoder. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 4352–4360.

[88] Elena Zheleva and Lise Getoor. 2009. To Join or Not to Join: The Illusion of Privacy in Social Networks with Mixed Public and Private User Profiles. In *Proceedings of the 18th International Conference on World Wide Web* (Madrid, Spain) *(WWW '09)*. Association for Computing Machinery, New York, NY, USA, 531–540. https://doi.org/10.1145/1526709.1526781

A APPENDIX

A.1 Theorem for Upper bound on $I(\mathbf{x_c};\mathbf{u})$

Theorem A.1. *Given a random vector $\mathbf{x} \in R^n$ with covariance matrix $\mathbf{K}$, then:*

$$\mathcal{H}(\mathbf{x}) \leq \frac{1}{2}\log((2\pi e)^n |\mathbf{K}|) \tag{8}$$

Proof. This theorem is proved using the fact that the KL-divergence of two distributions is always positive. The complete proof is in [15], Theorem 8.6.5. □

A.2 Lower bound on $I(\mathbf{x_c};\mathbf{c})$

First, we introduce a lemma [60] that we use for finding the lower bound of $I(\mathbf{x_c};\mathbf{c})$.

Lemma A.2. For any arbitrary conditional distribution $q(\mathbf{c}|\mathbf{x_c})$, we have:

$$\mathbb{E}_{\mathbf{x_c},\mathbf{c}}[\log \frac{q(\mathbf{c}|\mathbf{x_c})}{p(\mathbf{c})}] \leq I(\mathbf{x_c};\mathbf{c}) \tag{9}$$

Proof. Since we know that KL-divergence is always non-negative, we can write:

$$D_{KL}(p(\mathbf{c}|\mathbf{x_c})||q(\mathbf{c}|\mathbf{x_c})) = \int p(\mathbf{c}|\mathbf{x_c}) \log \frac{p(\mathbf{c}|\mathbf{x_c})}{q(\mathbf{c}|\mathbf{x_c})}\, d\mathbf{c} \geq 0$$

From this, we can come to:

$$\int p(\mathbf{c},\mathbf{x_c}) \log \frac{p(\mathbf{c}|\mathbf{x_c})p(\mathbf{c})}{q(\mathbf{c}|\mathbf{x_c})p(\mathbf{c})}\, d\mathbf{c}\, d\mathbf{x_c} \geq 0$$

By negation, we get:

$$-\int p(\mathbf{c},\mathbf{x_c}) \log \frac{p(\mathbf{c}|\mathbf{x_c})p(\mathbf{c})}{q(\mathbf{c}|\mathbf{x_c})p(\mathbf{c})}\, d\mathbf{c}\, d\mathbf{x_c} \leq 0 \tag{10}$$

On the other hand, from the definition of mutual information, we can write:

$$I(\mathbf{x_c};\mathbf{c}) = \int p(\mathbf{c},\mathbf{x_c}) \log \frac{p(\mathbf{c},\mathbf{x_c})}{p(\mathbf{c})p(\mathbf{x_c})}\, d\mathbf{x_c}\, d\mathbf{c} \tag{11}$$

If we add $I(\mathbf{x_c};\mathbf{c})$ from Equation 11 to 10, we get:

$$\int p(\mathbf{x_c},\mathbf{c}) \log \frac{q(\mathbf{c}|\mathbf{x_c})}{p(\mathbf{c})} \leq I(\mathbf{x_c};\mathbf{c})$$

Which yields:

$$\mathbb{E}_{\mathbf{x_c},\mathbf{c}}[\log \frac{q(\mathbf{c}|\mathbf{x_c})}{p(\mathbf{c})}] \leq I(\mathbf{x_c};\mathbf{c}) \tag{12}$$

□

Now, we review the theorem and prove it.

Theorem 3.2. *The lower bound on $I(\mathbf{x_c};\mathbf{c})$* is:

$$\mathcal{H}(\mathbf{c}) + \max_q \mathbb{E}_{\mathbf{x_c},\mathbf{c}}[\log q(\mathbf{c}|\mathbf{x_c})] \tag{13}$$

By q, we mean all members of a possible family of distributions for this conditional probability.

Proof. For all q, the left hand side of equation 9 offers a lower bound. The equality happens when $q(\mathbf{c}|\mathbf{x_c})$ is equal to $p(\mathbf{c}|\mathbf{x_c})$. Given this, if we estimate a close enough distribution q that maximizes the left hand side of the inequality 9, we can find a tight lower bound for the mutual information. We can re-write inequality 9 as:

$$-\mathbb{E}_{\mathbf{c}}[\log p(\mathbf{c})] + \mathbb{E}_{\mathbf{x_c},\mathbf{c}}[\log q(\mathbf{c}|\mathbf{x_c})] \leq I(\mathbf{x_c};\mathbf{c})$$

Based on the definition of Entropy and the discussion above about tightening the bound, the lower bound on the mutual information is:

$$\mathcal{H}(\mathbf{c}) + \max_q \mathbb{E}_{\mathbf{x_c},\mathbf{c}}[\log q(\mathbf{c}|\mathbf{x_c})] \tag{14}$$

□

A.3 Hyperparameters for Training

Tables 2, 3 and 4 show the hyperparameters used for training in the experiments of Sections 5.2, 5.3 and 5.4. For the first one, the *Point#* indicates the experiment that produced the given point in the graph, if the points were numbered from left to right. The hyperparameters of the rest of the experiments are the same as the ones brought. In our implementation, for ease of use and without loss of generality, we have introduced a variable γ to the loss function in Equation 7, in a way that $\gamma = \frac{1}{\lambda}$. With this introduction, we do not directly assign a λ (as if λ were removed and replaced by γ as a coefficient of the other term). In the tables, we have used lambda to be consistent, and in the cells where the value for λ is not given, it means that the loss is only cross-entropy. But in the Code, the coefficient is set on the other term and is $1/\lambda$s reported here. The batch sizes used for training are 128 for CIFAR-100, MNIST, and UTKFace and 40 and 30 for CelebA. For testing the batch size is 1, so as to sample a new noise tensor for each image and capture the stochasticity. Also, the number of samples taken for each update in optimization is 1 since we do mini-batch training and for each mini-batch we take a new sample. Finally, M is set to 1.5 for all benchmarks, except for CelebA where it is set to be 5.

A.4 Code Directory Structure

The code and model checkpoints used to produce the results are provided at https://github.com/mireshghallah/cloak-www-21. The code is in the directory `code` and the models and NumPy files are named `saved_nps.zip` and they both have the same directory structure. They each contain 5 Folders named *exp1-trade-off, exp2-adversary, exp3-black-box, exp4-fairness* and *exp5-shredder* which are related to the results in the experiments section in the same order. The pre-trained parameters needed are provided in the `saved_nps.zip`, in the corresponding directory. So, all that is needed to be done is to copy all files from the `saved_nps.zip` directory to their corresponding positions in the code folders, and then run the provided Jupyter notebooks. The notebooks that were used to generate representations are provided, in case someone wants to reproduce the results, and the saved Cloak models and pre-trained models are given as well. For acquiring the datasets, you can have a look at the `acquire_datasets.ipynb` notebook, included in the `code.zip`.

In short, each notebook has Cloak in its name will start by loading the required datasets and then creating a model. Then, the model is trained based on the experiments and using the hyperparameters provided in section A.3. Finally, you can run a test function that is provided to evaluate the model. For seeing how the mutual information is estimated, you can run the notebooks that have `mutual_info` in their names. You need not have run the training beforehand if you place the provided `.npy` files in the correct directories. For the mutual information estimation, you will need to download the ITE toolbox [79]. The link is provided in the code.

Table 2: hyper parameters for Section 5.2

Model	Point#	Training Phase 1			Training Phase 2		
		epoch	LR	λ	epoch	LR	λ
CIFAR-100	1	17	0.001	1	3	0.001	10
	2	24	0.001	1	2	0.001	10
	3	30	0.001	1	2	0.001	10
	4	40	0.001	0.2	2	0.001	10
	5	140	0.001	0.2	2	0.001	10
MNIST	1	50	0.01	100	90	0.001	200
	2	50	0.01	100	160	0.001	200
	3	50	0.01	100	180	0.001	200
	4	50	0.01	100	260	0.001	100
	5	50	0.01	100	290	0.001	100
UTKFace	1	6	0.01	0.1	6	0.0001	100
	2	4	0.01	0.1	2	0.0001	100
	3	8	0.01	0.1	2	0.0001	100
	4	10	0.01	0.1	2	0.0001	100
	5	12	0.01	0.1	2	0.0001	100

Table 3: hyper parameters for Section 5.4

Model	Point#	Training Phase 1			Training Phase 2			Training Phase 3		
		epoch	LR	λ	epoch	LR	λ	epoch	LR	λ
VGG16	1	0.5	0.01	1	0.5	0.001	1	-	-	-
	2	0.5	0.01	1	0.7	0.001	1	-	-	-
	3	0.5	0.01	1	0.8	0.001	1	-	-	-
	4	0.8	0.01	1	0.8	0.001	1	0.2	0.001	5
	5	1	0.01	1	0.8	0.001	1	0.2	0.001	100
ResNet18	1	1	0.01	10	0.5	0.001	1	-	-	-
	2	1	0.01	5	0.5	0.001	1	-	-	-
	3	1	0.01	5	0.7	0.001	1	-	-	-
	4	1.2	0.01	3	0.5	0.001	1	0.2	0.001	5
	5	2	0.01	5	0.5	0.001	1	0.2	0.001	5

Table 4: hyper parameters for Section 5.3

Model	SR(%)	Training Phase 1		Training Phase 2		Training Phase 3	
		epoch	LR	epoch	LR	epoch	LR
Adversary-hair	00.00	1	0.01	-	-	-	-
	33.60	1	0.01	2	0.0001	1	0.00001
	53.70	1	0.01	2	0.0001	1	0.00001
	71.00	1	0.01	2	0.0001	1	0.00001
	89.70	1	0.01	2	0.0001	3	0.00001
	95.60	1	0.01	2	0.0001	2	0.00001
	98.30	1	0.01	2	0.0001	3	0.00001
Adversary-glasses	00.00	1	0.01	-	-	-	-
	33.60	1	0.01	2	0.0001	1	0.00001
	53.70	1	0.01	2	0.0001	1	0.00001
	71.00	1	0.01	2	0.0001	1	0.00001
	89.70	1	0.01	2	0.0001	3	0.00001
	95.60	1	0.01	2	0.0001	2	0.00001
	98.30	1	0.01	2	0.0001	3	0.00001

A.5 Fairness Metrics

In a classification task, demographic parity requires the conditional probability of the classifier predicting output class $\hat{Y} = y$ given sensitive variable $S = 0$ to be the same as predicting class $\hat{Y} = y$ given $S = 1$. In other words, $P(\hat{Y} = y|S = 0) = P(\hat{Y} = y|S = 1)$. Since in most real cases these values are not the same, the maximum pair-wise difference between these values is considered as a measure of fairness, Δ_{DemP}, and the lower this difference, the more fair the classifier. Here S would be the gender, which due to the data provided in the dataset, is binary. We have only two target classes of black hair and non-black hair, so the $\Delta_{DemP}(y = 0)$ is the same as $\Delta_{DemP}(y = 1)$.

Equalized odds is another fairness measure, which requires the conditional probability of the classifier predicting class $\hat{Y} = y$ given sensitive variable $S = 0$ and ground truth class $Y = y$ be equal to the same conditional probability but with $S = 1$. In other words, $P(\hat{Y} = y|S = 0, Y = y) = P(\hat{Y} = y|S = 1, Y = y)$. Similar to the demographic parity case, we also measure the difference in these conditional probabilities for both $y = 1$ (black hair) and $y = 0$ (non-black hair) and report their summation as Δ_{EO}, commensurate with [50].