A Principled Approach to Learning Stochastic Representations for Privacy in Deep Neural Inference

Preprint · March 2020

CITATIONS
0

READS
188

6 authors, including:

Fatemehsadat Mireshghallah
University of California, San Diego
11 PUBLICATIONS **44** CITATIONS
SEE PROFILE

Mohammadkazem Taram
University of California, San Diego
13 PUBLICATIONS **116** CITATIONS
SEE PROFILE

Ali Jalali
Amazon
2 PUBLICATIONS **16** CITATIONS
SEE PROFILE

Dean M. Tullsen
University of California, San Diego
208 PUBLICATIONS **15,988** CITATIONS
SEE PROFILE

A Principled Approach to Learning Stochastic Representations for Privacy in Deep Neural Inference

Fatemehsadat Mireshghallah [1] **Mohammadkazem Taram** [1]
Ali Jalali [2] **Ahmed Taha Elthakeb** [1] **Dean Tullsen** [1] **Hadi Esmaeilzadeh** [1]

Abstract

INFerence-as-a-Service (INFaaS) in the cloud has enabled the prevalent use of Deep Neural Networks (DNNs) in home automation, targeted advertising, machine vision, etc. The cloud receives the inference request as a raw input, containing a rich set of private information, that can be misused or leaked, possibly inadvertently. This prevalent setting can compromise the privacy of users during the inference phase. This paper sets out to provide a principled approach, dubbed Cloak, that finds optimal stochastic perturbations to obfuscate the private data before it is sent to the cloud. To this end, Cloak reduces the information content of the transmitted data while conserving the essential pieces that enable the request to be serviced accurately. The key idea is formulating the discovery of this stochasticity as an *offline gradient-based optimization problem* that reformulates a pre-trained DNN (with optimized known weights) as an analytical function of the stochastic perturbations. Using Laplace distribution as a parametric model for the stochastic perturbations, Cloak learns the optimal parameters using *gradient descent* and Monte Carlo sampling. This set of optimized Laplace distributions further guarantee that the injected stochasticity satisfies the ϵ-differential privacy criterion. Experimental evaluations with real-world datasets show that, on average, the injected stochasticity can reduce the information content in the input data by 80.07%, while incurring 7.12% accuracy loss.

1. Introduction

The success of deep learning in many areas including vision, recommendation systems, natural language processing, etc., has heralded the adoption of Deep Neural Networks (DNNs) in production systems (Laine et al., 2017; Këpuska & Bohouta, 2018). However, the computational complexity of DNNs has pushed their execution to mostly cloud infrastructure, where an edge device on the user side captures and sends the raw inputs (requests) to the cloud for *inference*. This execution model, called INFerence-as-a-Service (INFaaS) (Soifer et al., 2019; La et al., 2015), has become de-facto (e.g., mobile phones), yet it poses serious privacy concerns (Shokri et al., 2017; Warzel, 2019).

The threat is that as soon as the raw data is sent to the cloud, it can be misused or leaked through security vulnerabilities even if the cloud provider and/or the communication link is trusted (Kocher et al., 2019; Thompson & Warzel, 2019; Lipp et al., 2018; Newcomb, 2018). Such a risk is present for every single request (input) and is exacerbated by the fact that the raw inputs contains a rich set of information that is not directly relevant to the inference task. This paper aims to devise Cloak, a systematic approach towards providing a differentially private inference mechanism by adding perturbations to the primary input (request). An unprincipled addition of perturbations will lead to significant loss in the inference accuracy, putting the service in the position of questionable utility.

To address these challenges, the proposed key idea is formulating the discovery of the perturbation as an *offline gradient-based optimization problem* that reformulates a pre-trained DNN (with optimized known weights) as an analytical function of the stochastic perturbations. Using Laplace distribution as a parametric model for the stochastic perturbations, Cloak learns the optimal parameters using *gradient descent* and Monte Carlo sampling. With this setup, these *learned* distributions can obfuscate the private data while retaining the cloud's ability to perform the inference task. This set of optimized Laplace distributions further guarantee that the learned perturbations meet the ϵ-differential privacy criterion. To meet ϵ (the desired privacy budget), we incorporate it as a hard constraint in Cloak's loss function to maximize the accuracy with respect to this constraint. Cloak takes a further step after achieving the highest accuracy for a given ϵ and explores an alternative space by reducing Mutual Information (MI) between the original input and its perturbed representation. This approach opens a trade-off between accuracy and information content of the perturbed input that is controlled using a Lagrange multiplier that acts as a knob.

It worth emphasizing that Cloak's learning process is offline and is not invoked during inference. For each inference request, a distinct random perturbation tensor is sampled

[1]Department of Computer Science and Engineering, University of California San Diego [2]Amazon Co.. Correspondence to: Fatemehsadat Mireshghallah <fmireshg@eng.ucsd.edu>.

from the learned distributions and is added to the raw input.

Cloak offers these capabilities with a stark contrast with recently emerging line of work on using noise for inference privacy that require to know what they are protecting against (i.e., a private label). Cloak, however, does not need this extra labeling and it attempts at removing any excessive information that is not conducive to the main inference task/label. Additionally, unlike these techniques, Cloak does not need to change parameters or the architecture of the pre-trained networks. Instead, it produces a significantly perturbed representation of the input, aiming to only provide just enough information to serve the inference task (Figure 1). Furthermore, the aforementioned prior works do not *directly* learn the perturbations (Author, 2020; Osia et al., 2020; Osia et al., 2020; Wang et al., 2018). Finally, Cloak does not impose the prohibitive computational cost of techniques that use homomorphic encryption (Dowlin et al., 2016; Gentry, 2009; Bos et al., 2013) which can increase the execution time of a neural network by three orders of magnitude (Riazi et al., 2019).

We provide an analytical formulation of the *stochasticity learning* problem as a constrained convex optimization program that maximizes the privacy by minimizing the mutual information between the raw input and the data sent to the cloud subject to a restriction on the degradation of the DNN utility (accuracy). Cloak's objective function and the constraint both depend on the prior distribution of the perturbation. However, for a fixed parameterized family of distributions and a fixed value of the Lagrange multiplier knob, the problem has a unique global optima. The convexity of the formulation guarantees that the *gradient descent* will converge to that global optima. This proves that Cloak maximizes the privacy under the utility preserving constraint. Further, we prove Cloak satisfies differential privacy guarantees with respect to the features of the input. The per-feature guarantee ensures that an adversary cannot use the spatio-temporal correlations to reverse engineer the process and gain access to the sensitive information.

Experimental evaluation with real-world datasets of UTKFace (Zhang & Qi, 2017), CIFAR-100 (Krizhevsky, 2009), and MNIST (LeCun & Cortes) shows that Cloak can reduce the mutual information content of input images by 80.07% with accuracy loss of 7.12% for an ϵ of 2.5. The large dimensions of images in the CelebA dataset prevent the accurate estimation of mutual information. As such, we take a practical approach towards evaluation of this dataset. First, we visualize the effects of Cloak pictorially on Figure 1. Then, we select the inference service as "smile detection" on this dataset. The results show that Cloak reduces the accuracy of a malevolent DNN classifier that aims to infer "gender," from 96.7% to 65.7%–reduction of 31.0%. This significant reduction is achieved while Cloak does not assume or incorporate anything about the label, feature, or sensitivity (privacy) of "gender." Additionally, Cloak preserves the accuracy of "smile detection" service at 87.2%–only 4.9% accuracy degradation. The code for the experiments is included in the submission and all the hyperparameters and experiment details are provided in the supplementary material.

2. Preliminaries

In this section, we discuss the notation and fundamental concepts used in the rest of the paper. We first review differential privacy and the Laplace mechanism, which is an ϵ-differentially private mechanism that Cloak is built upon. Then, we briefly review mutual information formulation, that is another conventional measure for privacy and is also used in this paper.

2.1. Function Notations

We define function f, to be the target function the output of which is to be perturbed. This could be any pre-processing (e.g., normalization) carried out on the raw input data, before being handed to the untrusted party. In particular, this function can be the identity function if there is no pre-processing steps. We also define function g to be the neural network computation which is to be executed by the untrusted party.

2.2. Differential Privacy

Definition 2.1. ϵ-Differential Privacy (ϵ-DP). For $\epsilon \geq 0$, an algorithm A satisfies ϵ-DP (Dwork et al., 2006a;b) if and only if for any pair of datasets D and D' that differ in only one element:

$$\mathcal{P}[A(D)=t] \leq e^{\epsilon}\mathcal{P}[A(D')=t] \ \forall t \tag{1}$$

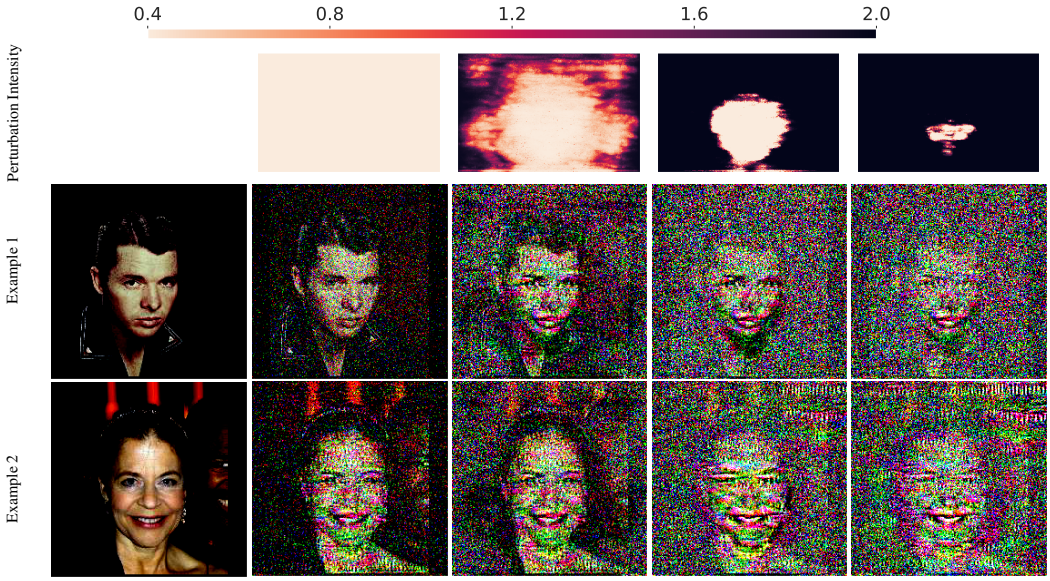
where, $\mathcal{P}[A(D)=t]$ denotes the probability that the algorithm A outputs t. DP tries to approximate the effect of an individual opting out of contributing to the dataset, by ensuring that any effect due to the inclusion of one's data is small.

2.3. Laplace Mechanism

Definition 2.2. Laplace Mechanism. (Dwork et al., 2006b) Given a target function f and a fixed $\epsilon \geq 0$, the randomizing algorithm $A_f(D) = f(D) + x$ where x is a perturbation random variable drawn from a Laplace distribution $Lap(\mu, \frac{\Delta_f}{\epsilon})$, is called the Laplace Mechanism and is ϵ-DP. Here, Δ_f is the global sensitivity of function f, and is defined as $\Delta_f = \sup|f(D) - f(D')|$ over the all dataset pairs (D,D') that differ in only one element.

2.4. Mutual Information

The amount of mutual information between the raw data and the representation that is to be publicized is another measure of privacy that is widely used in literature (Cuff & Yu, 2016; Kalantari et al., 2017; Liao et al., 2017). Cloak aims to minimize the mutual information between the raw input (D) and its perturbed representation, $A_f(D)$, that was attained via the randomization algorithm A_f. We can bound



(a) Original Image (b) $\bar{\mathsf{B}}=0.44$ (c) $\bar{\mathsf{B}}=1.01$ (d) $\bar{\mathsf{B}}=1.70$ (e) $\bar{\mathsf{B}}=1.95$

Figure 1. Visualizing the effect of Cloak on the input images as we increase the average scale of perturbations $\bar{\mathsf{B}}$ that is proportional to the standard deviation. The non-sensitive classification task is smile detection. The first row shows the heat map of perturbation scales for each pixel. At higher scales, Cloak obfuscates the features that are non-conducive to the smile detection task e.g. the background, the hair, etc., while only the lips and some minor facial attributes are recognizable.

this mutual information by

$$\begin{aligned}\mathcal{I}(D;A_f(D)) &\leq \mathcal{I}(f(D);A_f(D))\\ &= \mathcal{H}(A_f(D)) - \mathcal{H}(A_f(D)|f(D))\\ &= \mathcal{H}(A_f(D)) - \mathcal{H}(x) \qquad (2)\\ &\leq \mathcal{H}(f(D)) - \mathcal{H}(x)\\ &= \mathcal{H}(f(D)) - \log(2be)\end{aligned}$$

where, $\mathcal{H}(\cdot)$ represents the Shannon entropy (Shannon, 1948). Here, we assume x is distributed as $Lap(\mu,b)$, i.e., b is the scale of the Laplace distribution. The first inequality follows from the data processing inequality (Beaudry & Renner, 2011). The first term in the last line of Equation (2) depends on the target function f and is a constant with respect to the optimization variable x in our case. Hence, we do not consider it for our optimization. The second term can be written as $-\log(2be) = -\log(2e) - \log(b)$, where $-\log(2e)$ is constant. Thus, If we minimize $-\log(b)$, i.e., maximize the scale of the distribution of perturbations, we have minimized an upper bound on the mutual information of the leaked data.

3. Offline Training for Perturbed Inference

During the inference phase, for each input data tensor (D), Cloak adds a distinct randomly generated perturbation tensor (X). This addition yields a perturbed representation of the input that is sent out to the cloud for classification with the DNN denoted by g. During the inference phase no optimization is performed. X is generated by sampling each element, x, independently, from a Laplace distribution $Lap(\mu,b)$, where μ and b are the corresponding elements of the locations (M) and scales tensors (B).

In a separate *offline* process, Cloak finds these M and B tensors by solving an optimization problem. Cloak trains the locations and scales tensors in such a way that: (1) provides the required differential privacy guarantee, as defined shortly; (2) incurs minimum degradation to the neural network's objective, and, (3) decreases the amount of mutual information between the original tensor D and its noisy, perturbed version, $A_f(\mathsf{D})$. When the desired tensors M and B are found, they are used for perturbing input data for inference, in deployed neural networks as mentioned above.

This section first defines an instantiation of differential privacy, called feature differential privacy, that applies to our problem setup. The section continues on to discuss the constraints that need to be applied to the locations and scales tensors to meet the required guarantee and related privacy consideration. Finally, this section explain the workflow in details and provide a proof that the proposed mechanism is ϵ-feature differentially private.

3.1. ϵ-Feature Differential Privacy

The goal of differential privacy is to guarantee that one cannot infer whether an individual data instance was in the dataset by observing an algorithm's output. In Cloak's problem setting, we apply differential privacy to obfuscate personal/sensitive features, within each data instance. This problem can be looked upon as trying to make the changes in one feature within the data instance indistinguishable. As such, we define an instantiation of differential privacy,

where a data instance is considered as a dataset and features are considered records of this dataset, as *feature differential privacy*. For instance, in images, the features can be the pixels, and one attempt at protecting sensitive information in the image could be adding noise through a deferentially private mechanism, so as to make pixels less distinguishable. In this case, the datasets D and D' can be two identical images that only differ in one pixel.

3.2. Noise Parameter Constraints

Cloak aims at casting the noise distribution parameters as a trainable tensor, and using conventional gradient based methods to train them. To be able to define gradients over the locations and scales, we rewrite the noise sampling to be $\mathsf{X} = \mathsf{B} \circ \mathsf{E} + \mathsf{M}$, instead of $\mathsf{X} \sim Lap(\mathsf{M},\mathsf{B})$, where E is a tensor with the same shape as the data tensor (D) and is sampled i.i.d from $Lap(0,1)$ and $\circ$ is the element-wise multiplication. This redefinition enables us to formulate the problem as an analytical function for which we can calculate the gradients.

To achieve the differential privacy guarantee, the elements of B must be larger than $\frac{\Delta_f}{\epsilon}$. We define a to-be-tuned hyper-parameter M_x as an upper bound and reparameterize the scales tensor B as P:

$$\mathsf{B} = \frac{1.0 + \tanh(\mathsf{P})}{2}(M_x - \frac{\Delta_f}{\epsilon}) + \frac{\Delta_f}{\epsilon} \tag{3}$$

As discussed experimentally later, M_x is usually between 1.5 to 2.5. We put this extra constraint on the distributions, so that none of the noise elements gets a very high scale and becomes an outlier compared to the scale of other noise elements.

3.3. Average and Worst-Case Privacy Considerations

Differential privacy offers a worst-case privacy guarantee. Cloak, however, decreases the mutual information between the input data and its perturbed representation, while maintaining the desired differential privacy guarantee. The reduction in mutual information can be considered as an average-case privacy measure (Huang et al., 2018; Liao et al., 2017).

As discussed, the mutual information between tensors D and $A_f(\mathsf{D})$ where the latter is acquired by injecting noise to $f(\mathsf{D})$, is proportional to the log of the standard deviation of the noise that is added to each feature (element) in $f(\mathsf{D})$. Therefore, Cloak needs to maximize the scales of the noise distributions to minimize the leaked mutual information.

Increasing the scale parameters of noise elements through training must be done in consideration of the neural network's objective. For instance, in a setting where the task is smile detection, some facial features like the mouth are conducive to this detection. Other features, however, such as that person's hair, the background, their eye color, etc. are irrelevant. Thus, adding more noise to these features can achieve higher privacy in terms of information loss, while incurring minimum degradation to the utility.

An example of noise injection can be seen in Figure 1, where the effect of increase in the average of the noise scales is illustrated for two sample images from the CelebA (Liu et al., 2015) dataset. In this illustration, Cloak causes non-conducive features to get blurry by increasing the scales. For the rightmost images, only the smile and some minor facial attributes are present. For instance, notice the second example image where the other person in the background fades away as the noise increases. The mean accuracy drop over the test portion of the dataset for the four perturbation distribution sets (four sets of tensors B and M) in this Figure are 3.4%, 3.7%, 4.5% and 12.7% respectively, with standard deviations in the $0.12\% - 0.68\%$ range. The privacy budget (ϵ) is 2.5.

Such a rather targeted (non-uniform) obfuscation behavior that leads to low degradation in accuracy for a given ϵ is the result of our formulation that incorporates mutual information in Cloak's loss function (see Section 3.4). If only the differential privacy constraint was considered, the noise would have just been increased uniformly across the image and accuracy loss would have been much higher. This behavior will be discussed in more details in Section 4.2.

3.4. Optimization Problem Formulation

Cloak's objective function is to minimize the mutual information (or maximize the noise variance as discussed before) subject to a bounded degradation in the neural network's utility. Using Lagrange multiplier λ and given our reparameterization of B, we can express the loss function of Cloak as

$$\mathcal{L}(\mathsf{M},\mathsf{B}) = -\log\mathsf{B} + \lambda\mathcal{L}_{nn}(\mathsf{M},\mathsf{B}) \tag{4}$$

where, $\mathcal{L}_{nn}$ represents the utility loss function of the neural network. For instance, in the case of C-class classification, this utility can be represented as $\mathcal{L}_{nn}(\mathsf{M},\mathsf{B}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_c^i \log(p_c^i)$, which is the cross entropy loss, where p_c^i is the i^{th} Monte Carlo sample (Kalos & Whitlock, 1986) of the probability the neural network has assigned to observation i belonging to class c, and, y_c^i is the indicator function of whether the i^{th} sample belongs to the class c. Converting the averaging over n samples into the population expectation, we can rewrite our loss function as

$$\begin{aligned}\mathcal{L}(\mathsf{M},\mathsf{B}) &= -\log\mathsf{B} \\ &+ \lambda\mathbb{E}_{\mathsf{X}\sim L(\mathsf{M},\mathsf{B})}\Big[\sum_{c=1}^{C} y_c \log\big(g\big(f(\mathsf{D}) + \mathsf{X}\big)_c\big)\Big]\end{aligned} \tag{5}$$

We take sufficiently large n noise samples drawn from the current locations and scales tensors to approximate the second term. This means that to apply a single update to the trainable parameters, Cloak runs multiple forward passes on the entire neural network, at each pass draws new samples for the noise (perturbation) tensor, and averages over the losses and applies the update using the average. However, in practice, if mini-batch training is used, we observed that using only a single noise sample for each update can yield desirable results,

Algorithm 1 Cloak's Perturbation Training Workflow

Input: $\mathsf{D}, y, f, \epsilon, \lambda$
Initialize $\mathsf{M}=0$, $\mathsf{P}=-\infty$ and $M_x > \frac{\Delta_f}{\epsilon}$
repeat
  Sample $\mathsf{E} \sim Lap(0,1)$
  Let $\mathsf{B}=\frac{1.0+\tanh(\mathsf{P})}{2}(M_x-\frac{\Delta_f}{\epsilon})+\frac{\Delta_f}{\epsilon}$
  Let $\mathsf{X}=\mathsf{B}\circ\mathsf{E}+\mathsf{M}$
  Take gradient step on M and P from Equation (4)
until Algorithm converges
Return: M, B

since a new noise tensor is sampled for each mini-batch.

The parameter λ is the knob, which could be tuned to reflect whether we want to trade-off accuracy for information degradation (more privacy), or vice versa. We name the $-log\mathsf{B}$, the mutual information decay term. One important feature of Cloak is that it does not need access to sensitive information/labels. In other words, it does not need to be given specific information about what it is protecting against, it only needs access to the non-sensitive labels of the main task, and it tries to remove any extra information with the trained perturbations. For more details of the optimization problem please refer to the supplementary material.

3.5. Cloak's Workflow

Cloak's optimization process can be seen in Algorithm 1. Before beginning the process, the locations tensor (M) is initialized to 0, and tensor P is initialized in such a way that the scales take the $(\frac{\Delta_f}{\epsilon})$ value. The rest of the parameters of the neural network are set to their pre-trained values. The functions f and g are the target function and neural network function, as defined in Section 2.1.

Once the training is finished, the optimized locations and scales are saved. During deployment (inference), for each input that is supposed to be given to the inference service, a set of perturbations is sampled from the optimized distributions. Each element in the perturbations tensor X is sampled independently and added to input before it is uploaded for inference.

3.6. Differential Privacy Proof

In this section we show that Cloak is ϵ-feature Differentially Private (DP), as defined in Section 3.1. We start by showing Cloak is ϵ-DP with respect to a single feature. We then show that Cloak is ϵ-feature DP, with respect to all features.

Proposition 3.1. Cloak is ϵ-DP w.r.t. a single feature.

Proof. For a single feature D_i in the dataset D, we have $A_f(D_i)=f(D_i)+\mathsf{X}_i$, where X_i is drawn from $Lap(\mathsf{M}_i,\mathsf{B}_i)$. This is a Laplace Mechanism and hence it is ϵ-DP. □

Proposition 3.2. Cloak is ϵ-feature differentially private with respect to all features.

Proof. By Proposition 3.1, each single feature i is ϵ_i-DP for some $\epsilon_i \geq 0$. Assume D and D' are two datasets that are different only on j^{th} feature. Since Cloak samples each noise perturbation independently, we have

$$\begin{aligned}\mathcal{P}[A_f(D)=t] &= \prod_i \mathcal{P}[A_i(D_i)=t_i] \\ &= \mathcal{P}[A_j(D'_j)=t_j]\prod_{i\neq j}\mathcal{P}[A_i(D'_i)=t_i] \\ &\leq e^{\epsilon_j}\mathcal{P}[A_j(D'_j)=t_j]\prod_{i\neq j}\mathcal{P}[A_i(D'_i)=t_i] \\ &\leq e^{\max(\epsilon_i)}\mathcal{P}[A(D')=t].\end{aligned}$$

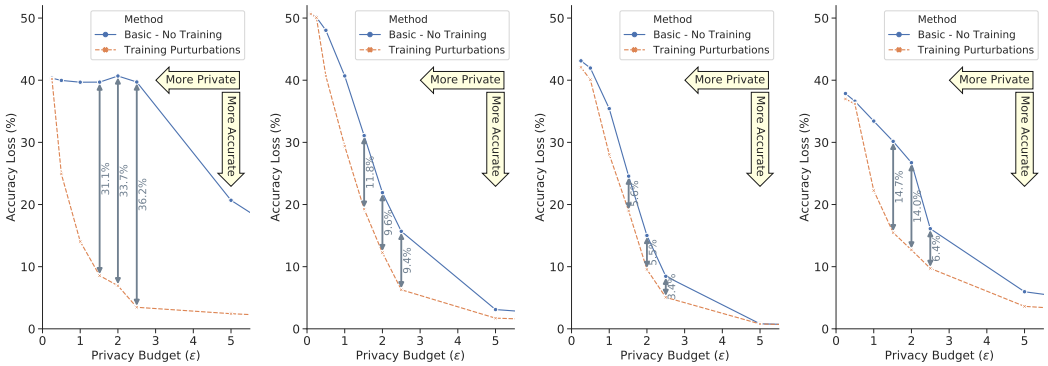
Hence, Cloak is ϵ-feature DP with respect to all features. □

4. Experimental Results

To evaluate Cloak, we use four real-world datasets on four DNNs. Namely, we use VGG-16 (Simonyan & Zisserman, 2014) on CelebA (Liu et al., 2015), AlexNet (Krizhevsky et al., 2012) on CIFAR-100 (Krizhevsky, 2009), a modified version of VGG-16 model on UTKFace (Zhang & Qi, 2017), and LeNet-5 (LeCun, 1998) on MNIST (LeCun & Cortes).

We define a set of non-sensitive tasks as inference services over these datasets. Specifically, we use smile 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. 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%. 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 inference 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 1.5%) the confidence bars are not visible on the graphs.

The input image sizes for CelebA, CIFAR-100, UTKFace. and MNIST are $224\times224\times3$, $32\times32\times3$, $32\times32\times3$, and 32×32, respectively. In addition, in our experiments, the inputs are all normalized to 1. In all of the experiments, we add the perturbations directly to the input image and create a noisy representation (similar to those in Figure 1) which we then be given to the neural network. Therefore, the function f defined in Section 2.1 is the identity function and the sensitivity Δ_f is 1. The experiments are all carried out using Python 3.6 and PyTorch 1.3.1. We use Adam optimizer (Kingma & Ba, 2015) for perturbation training. 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 (Szabó, 2014).



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

Figure 2. Accuracy loss vs. the privacy budget (ϵ) for the four benchmark datasets, measured on test set. *Basic - No Training* represents the results of adding random differentially-private perturbation. *Training Purturbations* shows the effect of training the perturbation distribution parameters. The neural network parameters remain frozen at all times. The gaps shown with vertical arrows indicate the improvement in accuracy brought by Cloak's perturbation training for a given ϵ.

Code and more information about the model architectures, mutual information estimation, the hyper-parameters used in each of the experiments, and experimental setup is provided in the supplementary material.

In the rest of this section, first, we cover the experiments that show, in detail, how different parts of the formulation contribute to utility and privacy. After that, we present an experiment to show how Cloak is performing in protecting sensitive features from malevolent classifier that tries to infer sensitive information from the input.

4.1. Effects of Learning Perturbations

Figure 2 shows the mean accuracy loss for a given privacy budget ϵ. Two methods are compared here. First, the *Basic - No Training* method that we consider as the baseline in which perturbation elements (the noise for each pixel) are independently generated from a Laplace distribution with location of zero and scale of $\frac{\Delta_f}{\epsilon}$. $\frac{\Delta_f}{\epsilon}$ is the minimum scale that the perturbation distributions can have, in order to achieve a certain privacy budget of ϵ for the feature-differential privacy criterion of Section 3.1.

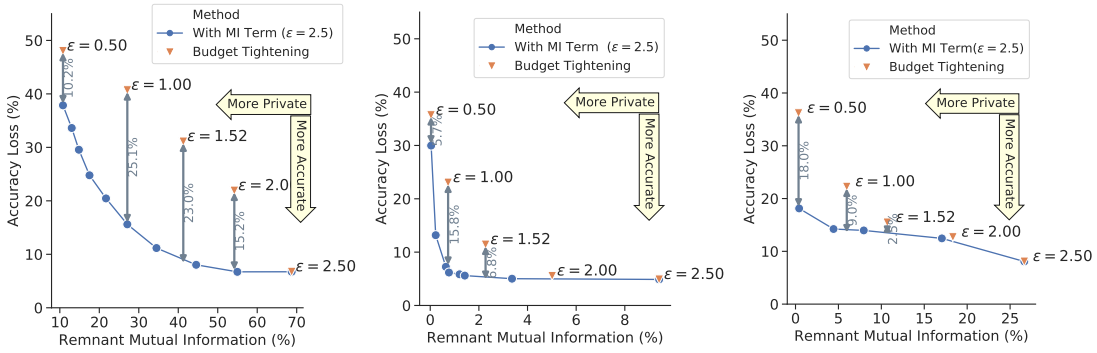
For the second method, Cloak is used to observe the effect of perturbation distribution training on the accuracy. To do so we removed the mutual information term from Equation 4. The trainable parameters are only the locations and scales (tensors M and B from Section 3) of perturbation distributions. This means that the number of trainable parameters are $2 \times input\_feature\_dimensions$, and the rest of the neural network parameters are frozen.

We observe that the scales (B) of the perturbation elements do not change during training, since Cloak is only optimizing for accuracy and is not trying to decrease the mutual information. So, the scales remain at their $\frac{\Delta_f}{\epsilon}$ value that delivers the ϵ-feature differential privacy guarantee. This implies that the improvement over the *No Training* method is merely caused by the training of the elements of the locations tensor. The vertical arrows in the graphs indicate the amount of accuracy improvement achieved through the perturbation training. For a budgets ϵ of less than 0.25, the amount of noise is so high that the perturbation training can only modestly mitigate the situation and all the methods yield similar results. Figure 2 is depicted for ϵ below 5, to zoom into the region of interest and show the details. We have included graphs with ϵ of up to 10 in the supplementary materials.

4.2. Effects of Mutual Information Term

Figure 3 shows the accuracy loss vs. the remnant mutual information in the original image. Remnant mutual information denotes the mutual information between the original image and its noisy representation, divided by the amount of information in bits in the original image. For the *With MI Term* method, the λ in Equation 4 is tuned, so that as the accuracy grows, the mutual information in the original images would degrade. The privacy budget is $\epsilon = 2.5$ for all the points depicted for this method, since the mutual information degradation does not violate the differential privacy guarantee by pushing any scale below $\frac{\Delta_f}{\epsilon}$ (Section 3.2).

The *Budget Tightening* method shows how much accuracy loss would be incurred, if the same level of mutual information is to be achieved through merely tightening the differential privacy guarantee, using Cloak without the MI term ($-log(\mathsf{B})$), similar to the training method of the previous section. The graphs show that using the MI term helps achieve a certain level of mutual information with less damage to model utility. The vertical arrows show the difference in utility (accuracy) for these two methods. The worse utility brought by naïvly tightening the differential privacy budget is because DP tries to make all the features in the image similar,



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

Figure 3. Accuracy loss vs. the remaining mutual information for a fixed differential-privacy budget of $\epsilon = 2.5$. The *Budget Tightening* method degrades the accuracy more for a given remnant mutual information level. The *With MI Term* method exploits the mutual information term in the loss to reach the desired mutual information level, without tightening the differential privacy guarantee by reducing ϵ. The gaps shown by the vertical arrows indicates the difference in utility of these approaches for a given level of mutual information reduction.

and adds perturbations with the same scale to all the pixels.

For both cases, the graphs show a trade-off between accuracy and the loss in mutual information. For CIFAR-100, since the task is classifying the 20 superclasses, there is inherently more need for information for classification, compared to the other benchmarks. Therefore, for a given level of accuracy loss, the information content cannot be decreased as much as it can in the other benchmarks. We do not present any numerical results for the CelebA dataset here, since the input images have an extremely large number of features and the mutual information estimator tool is not capable of estimating the mutual information.

4.3. Malevolent Setting to Infer Private Information

To evaluate the obfuscation that Cloak provides, we devise an experiment in which a malevolent party tries to infer a sensitive private label from an input that is sent to an inference service. To this end, we use CelebA dataset and use gender as the sensitive label while the inference service is set to detect smiles from the perturbed inputs. Cloak does not access to the private labels while learning the perturbations, and it only optimizes for the accuracy of the smile detection task, and decreasing the mutual information.

Figure 4 shows the results of this experiment. The malevolent party is assumed to have a pre-trained VGG-16 neural network for gender classification, that has an accuracy of 96.7% on the test set, shown by the *Gender Classification Baseline* line. Cloak trains perturbation parameters with $\epsilon = 2.5$, but with different levels of inter-element intensity, i.e., it uses mutual information term, similar to the ones depicted in Figure 1. The *Noisy Gender Classification* depicts the accuracy achieved by the malevolent party's neural network, when it receives the perturbed representations (of the test set) and runs inference on them. The *Noisy Gender Classification with Re-training* depicts the same accuracy, but when the malevolent neural network is re-trained using the perturbed representations of the training dataset.

We first assume a malevolent party that would re-train its entire neural network (mark all parameters as trainable), but it would not learn anything. We then assume a malevolent party that would limit the training to the last fully connected layer, which yielded the results seen under the *Noisy Gender Classification with Re-training*. We believe the reason that limiting the number of parameters is helpful is that it limits the fluctuations in the network, by fixing a huge portion of the model parameters. The last line in the Figure is the accuracy that a random predictor would provide, which is 50%. For smile detection with accuracy of 86.9%, the gender classifier suffers an accuracy loss of 30.7% and does 15.8% better than a random classifier. The drop in gender classification accuracy, and the disability in learning and gaining higher accuracy is caused by the perturbation obfuscating unrelated features. However, since the nonsensitive task is smile detection, there could be some features that both gender detection and smile detection rely on (like the mouth area). This means it might not be possible to completely remove gender information while still maintaining smile detection accuracy.

5. Related Work

Privacy preserving research can be broadly categorized along two dimensions: first, the phase on which they focus, i.e., training vs inference, and second, the method that they use, i.e., perturbation-based vs cryptography-based. The majority of these studies fall under the training category where they try to protect users' privacy whiling building ML models (Shokri & Shmatikov, 2015; Abadi et al., 2016; Jiang et al., 2013; Chaudhuri et al., 2009; 2013) or aggregating/publishing detests (Dwork et al., 2006b;a; Dwork & Roth, 2014). However, the impending importance of inference privacy has led to the emergence of recent research efforts in this direction (Osia et al., 2020; Osia et al., 2020; Wang et al., 2018; Dowlin et al., 2016; Leroux et al., 2018). These

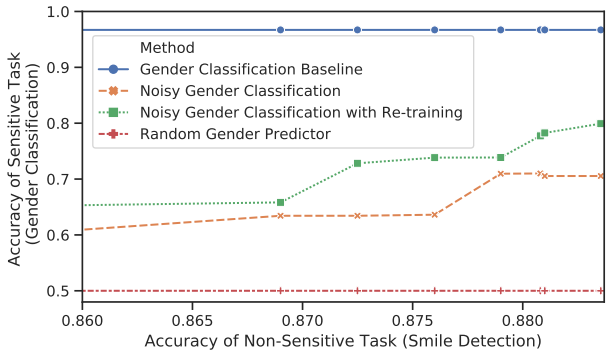


Figure 4. Cloak learns perturbations for the inference service of smile detection on CelebA dataset with 4.9% degradation to the accuracy of this service. If a malevolent party attempts to use the same perturbed image to infer sensitive (private) information, e.g. gender, the performance of this new inference service will be degraded by at least 31% while originally the accuracy was 96.7%. This shows Cloak is protecting the user against malicious use of their private information.

methods, unlike Cloak, either use homomorphic encryption that suffers from prohibitive computational costs (Dowlin et al., 2016), need labels for private information (Osia et al., 2020), or require significant changes/retraining in model architecture/weights (Wang et al., 2018). Additionally, in contrast to Cloak, these efforts do not formulate the problem as a direct analytical function of the perturbations that can be optimized using gradient descent. Below, the most related works are discussed in more details.

5.1. Noise-based Privacy Protection

For *training*, the literature abounds with the studies that use noise addition as a randomization mechanism to protect privacy (Chaudhuri et al., 2013; Dwork & Roth, 2014). Most notably, differential privacy (Dwork et al., 2006b), 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 (Chaudhuri & Monteleoni, 2009), statistical risk minimization (Chaudhuri et al., 2009), principal component analysis (Jiang et al., 2013; Chaudhuri et al., 2013), and deep learning (Shokri & Shmatikov, 2015; Abadi et al., 2016), to name a few. Nearly all of these studies have applied differential privacy to a training setting where they mainly concern with leaking private information in training set through the machine learning model. This paper, however, offers a differential private mechanism for DNN *inference*.

Only a handful of studies have addressed privacy of inference by adding noise to the data. Osia et al. (Osia et al., 2020) employed dimensionality reduction techniques such as principal component analysis (PCA) to reduce the amount of information before sending to untrusted cloud. Wang et al. (Wang et al., 2018) 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. (Leroux et al., 2018) propose an autoencoder to randomize the data, but the intensity of their obfuscation is too small to be irreversible, as they state. In a recent work (Author, 2020), we propose , to *heuristically* learn multiplicative and additive noise patterns by repeatedly collecting feasible patterns and fitting a distribution to them. In contrast, we formulate the problem of learning the distribution of the noise as a convex optimization that is directly and optimally solved through stochastic gradient descent. Due to the heuristic and pattern-based nature of this prior work, it cannot provide formal proofs nor does it satisfies the differential privacy criterion. More importantly, a significant part of its mutual information reduction comes from 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 the edge device cannot run the compute heavy beginning convolution layers of the neural network.

5.2. Cryptographic-based Privacy Protection

Privacy on offloaded computation can be provided by the means of cryptographic tools such as homomorphic encryption and/or Secure Multiparty Computation (SMC). Homomorphic encryption (Gentry, 2009; Bos et al., 2013) is a cyrptographic technique that allows deep learning operations to be performed in an encrypted domain (Dowlin et al., 2016). As such, there is no need for decryption, enabling the user to send ecrypted data to untrusted cloud without sharing the cryptographic key. However, these approaches suffer from a prohibitive computational costs, on both cloud and user side, exacerbating the complexity and compute-intensivity of neural networks especially on resource-constrained edge devices. Cloak in contrast avoids the significant cost of encryption and homomorphoic data processing.

Several other research (Tramer & Boneh, 2019; Hanzlik et al., 2018) rely on trusted execution environments such as Intel SGX and ARM TrustZone to run machine learning algorithm in remote hosts. However, this model requires the users to send their data to an enclave running on remote servers. In contrast to Cloak, this model still allows the remote server to have access to the raw data and as the new breaches in hardware (Kocher et al., 2019; Lipp et al., 2018) show, the access can lead to comprised privacy.

6. Conclusion

The surge in the use of machine learning is due to the growth in data and compute. The data mostly comes from people (Thompson & Warzel, 2019) and includes an abundance of sensitive information. We propose Cloak, a mechanism that learns differentially-private stochasticities that can help preserve privacy by obfuscating sensitive information in the raw input data while maintaining accuracy

for the intended inference in deep neural networks.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308–318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145/2976749.2978318.

Author, N. N. Suppressed for anonymity, 2020.

Beaudry, N. J. and Renner, R. An intuitive proof of the data processing inequality, 2011.

Bos, J. W., Lauter, K., Loftus, J., and Naehrig, M. Improved security for a ring-based fully homomorphic encryption scheme. In *Proceedings of the 14th IMA International Conference on Cryptography and Coding - Volume 8308*, IMACC 2013, pp. 45–64, Berlin, Heidelberg, 2013. Springer-Verlag. ISBN 978-3-642-45238-3. doi: 10.1007/978-3-642-45239-0_4. URL https://doi.org/10.1007/978-3-642-45239-0_4.

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

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. Differentially private empirical risk minimization, 2009.

Chaudhuri, K., Sarwate, A. D., and Sinha, K. A near-optimal algorithm for differentially-private principal components. *J. Mach. Learn. Res.*, 14(1):2905–2943, January 2013. ISSN 1532-4435.

Cuff, P. W. and Yu, L. Differential privacy as a mutual information constraint. In *CCS '16*, 2016.

Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 201–210. JMLR.org, 2016. URL http://dl.acm.org/citation.cfm?id=3045390.3045413.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9:211–407, August 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL http://dx.doi.org/10.1561/0400000042.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, pp. 486–503, Berlin, Heidelberg, 2006a. Springer-Verlag. ISBN 3-540-34546-9, 978-3-540-34546-6. doi: 10.1007/11761679_29. URL http://dx.doi.org/10.1007/11761679_29.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pp. 265–284, Berlin, Heidelberg, 2006b. Springer-Verlag. ISBN 3-540-32731-2, 978-3-540-32731-8. doi: 10.1007/11681878_14. URL http://dx.doi.org/10.1007/11681878_14.

Gentry, C. Fully homomorphic encryption using ideal lattices. In *In Proc. STOC*, pp. 169–178, 2009.

Hanzlik, L., Zhang, Y., Grosse, K., Salem, A., Augustin, M., Backes, M., and Fritz, M. Mlcapsule: Guarded offline deployment of machine learning as a service, 2018.

Huang, C., Kairouz, P., Chen, X., Sankar, L., and Rajagopal, R. Generative adversarial privacy. *CoRR*, abs/1807.05306, 2018. URL http://arxiv.org/abs/1807.05306.

Jiang, X., Ji, Z., Wang, S., Mohammed, N., Cheng, S., and Ohno-Machado, L. Differential-private data publishing through component analysis. *Transactions on data privacy*, 6(1):19, 2013.

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

Kalos, M. H. and Whitlock, P. A. *Monte Carlo Methods. Vol. 1: Basics*. Wiley-Interscience, USA, 1986. ISBN 0471898392.

Këpuska, V. and Bohouta, G. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 99–103, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Kocher, P., Horn, J., Fogh, A., , Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., and Yarom, Y. Spectre attacks: Exploiting speculative execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60:84–90, 2012.

La, H. J., Kim, M. K., and Kim, S. D. A personal healthcare system with inference-as-a-service. In *2015 IEEE International Conference on Services Computing*, pp. 249–255, June 2015. doi: 10.1109/SCC.2015.42.

Laine, S., Karras, T., Aila, T., Herva, A., Saito, S., Yu, R., Li, H., and Lehtinen, J. Production-level facial performance capture using deep convolutional neural networks. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350914. doi: 10.1145/3099564.3099581. URL https://doi.org/10.1145/3099564.3099581.

LeCun, Y. Gradient-based learning applied to document recognition. 1998.

LeCun, Y. and Cortes, C. The mnist dataset of handwritten digits. online accessed May 2019 http://www.pymvpa.org/datadb/mnist.html.

Leroux, S., Verbelen, T., Simoens, P., and Dhoedt, B. Privacy aware offloading of deep neural networks, 2018.

Liao, J., Kosut, O., Sankar, L., and Calmon, F. P. A general framework for information leakage. 2017.

Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., Horn, J., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., and Hamburg, M. Meltdown: Reading kernel memory from user space. In *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

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

Osia, S. A., Shamsabadi, A. S., Sajadmanesh, S., Taheri, A., Katevas, K., Rabiee, H. R., Lane, N. D., and Haddadi, H. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal*, pp. 1–1, 2020. ISSN 2372-2541. doi: 10.1109/JIOT.2020.2967734.

Osia, S. A., Taheri, A., Shamsabadi, A. S., Katevas, K., Haddadi, H., and Rabiee, H. R. Deep private-feature extraction. *IEEE Transactions on Knowledge and Data Engineering*, 32(1):54–66, Jan 2020. ISSN 2326-3865. doi: 10.1109/tkde.2018.2878698. URL http://dx.doi.org/10.1109/TKDE.2018.2878698.

Riazi, M. S., Samragh, M., Chen, H., Laine, K., Lauter, K., and Koushanfar, F. Xonn: Xnor-based oblivious deep neural network inference. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, pp. 1501–1518, USA, 2019. USENIX Association. ISBN 9781939133069.

Shannon, C. E. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

Shokri, R. and Shmatikov, V. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pp. 1310–1321, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. doi: 10.1145/2810103.2813687. URL https://doi.org/10.1145/2810103.2813687.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, May 2017. doi: 10.1109/SP.2017.41.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Soifer, J., Li, J., Li, M., Zhu, J., Li, Y., He, Y., Zheng, E., Oltean, A., Mosyak, M., Barnes, C., Liu, T., and Wang, J. Deep learning inference service at microsoft. In *2019 USENIX Conference on Operational Machine Learning (OpML 19)*, pp. 15–17, Santa Clara, CA, May 2019. USENIX Association. ISBN 978-1-939133-00-7. URL https://www.usenix.org/conference/opml19/presentation/soifer.

Szabó, Z. Information theoretical estimators toolbox. *Journal of Machine Learning Research*, 15:283–287, 2014.

Thompson, S. A. and Warzel, C. The privacy project: Twelve million phones, one dataset, zero privacy, 2019. online accessed February 2020 https://www.nytimes.com/interactive/2019/12/19/opinion/location-tracking-cell-phone.html.

Tramer, F. and Boneh, D. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJVorjCcKQ.

Wang, J., Zhang, J., Bao, W., Zhu, X., Cao, B., and Yu, P. S. Not just privacy. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2018. doi: 10.1145/3219819.3220106. URL http://dx.doi.org/10.1145/3219819.3220106.

Warzel, C. The privacy project: Faceapp shows we care about privacy but don't understand it, 2019. online accessed February 2020 https://www.nytimes.com/2019/07/18/opinion/faceapp-privacy.html.

Zhang, Zhifei, S. Y. and Qi, H. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

A. Appendix

A.1. Formal Problem Representation

Deep Neural Network: Consider a deterministic pre-processing function $f(\mathsf{D})$ that operates on the input tensor data D. We can model a DNN by a deterministic function $g(f(\mathsf{D});\theta^*)$, where, $f(\mathsf{D})$ is the tensor of pre-processed input data and θ^* is the fixed optimized parameter that fully determines the function g (and maximizes its utility over the distribution of the input tensor). We assume the DNN is trained and the parameter θ^* remains the same throughout the process. As such, for the ease of the notation, we might drop the parameter θ^*.

Performance: Suppose $\mathcal{L}_{nn}(\mathsf{D},g(f(\mathsf{D});\theta)$ is a loss function such that

$$\theta^* \in \operatorname{argmin}_{\theta} \mathcal{L}_{nn}(\mathsf{D},g(f(\mathsf{D});\theta))$$

If $\mathcal{L}_{nn}$ is the loss function corresponding to the DNN, then we are assuming that the network is trained and θ^* is the optimal parameter.

Privacy: We define the privacy P as the negative of the mutual information between the input of the function g and the raw data D, i.e.,

$$P = -\mathcal{I}(\mathsf{D};f(\mathsf{D}))$$

where $\mathcal{I}(\cdot;\cdot)$ represents mutual information. The lower mutual information implies higher privacy. Since f is a deterministic function, we have

$$P = -\mathcal{H}(f(\mathsf{D})) + \mathcal{H}\big(f(\mathsf{D})\big|\mathsf{D}\big) = -\mathcal{H}(f(\mathsf{D})) \tag{6}$$

where $\mathcal{H}(\cdot)$ denotes the entropy function.

Noise Injection: Denote a perturbation tensor by X which is independent of D. We intend to perturb the input to function g by changing $f(\mathsf{D})$ to $f(\mathsf{D})+\mathsf{X}$, i.e., the output of function g changes from $g(f(\mathsf{D});\theta^*)$ to $g(f(\mathsf{D})+\mathsf{X};\theta^*)$. It is worth reemphasizing that the parameter θ^* remains unchanged. Now, we need to change the privacy measure P to $\hat{P}$ where $\hat{P} = -\mathcal{I}(\mathsf{D};g(f(\mathsf{D})+\mathsf{X}))$. We can provide the following lower bound

$$\begin{aligned}\hat{P} &= -\mathcal{I}(\mathsf{D};f(\mathsf{D})+\mathsf{X}) \\ &\geq -\mathcal{I}(f(\mathsf{D});f(\mathsf{D})+\mathsf{X}) \\ &= -\mathcal{H}(f(\mathsf{D})) + \mathcal{H}\big(f(\mathsf{D})\big|f(\mathsf{D})+\mathsf{X}\big) \\ &= P + \mathcal{H}\big(f(\mathsf{D})\big|f(\mathsf{D})+\mathsf{X}\big)\end{aligned} \tag{7}$$

where the last equality is derived from (6). This equation implies that by noise injection process, we can improve the privacy at least by $\mathcal{H}\big(f(\mathsf{D})\big|f(\mathsf{D})+\mathsf{X}\big)$.

Optimization Problem: After noise injection, we would like to find the optimal perturbation X^* that maximizes the privacy while keeping the performance at an admissible error range. In other words, we are looking for the solution to the following optimization problem

$$\mathsf{X}^* = \operatorname{argmax}_{\mathsf{X}} \hat{P} \;\; \mathbf{s.t.} \;\; \mathcal{L}_{nn}(\mathsf{D},g(f(\mathsf{D})+\mathsf{X};\theta^*)) \leq \gamma$$

Given (7), we use $\mathcal{I}\left(f(\mathsf{D});f(\mathsf{D})+\mathsf{X}\right) = \mathcal{H}(f(\mathsf{D})+\mathsf{X}) - \mathcal{H}\big(f(\mathsf{D})+\mathsf{X}\big|f(\mathsf{D})\big) = \mathcal{H}(f(\mathsf{D})+\mathsf{X}) - \mathcal{H}(\mathsf{X}))$ as a surrogate objective and reformulate the problem in terms of a Lagrange multiplier λ as

$$\begin{aligned}\mathsf{X}^* = \operatorname{argmin}_{\mathsf{X}} \; &\mathcal{H}(f(\mathsf{D})+\mathsf{X}) - \mathcal{H}(\mathsf{X})) \\ &+ \lambda\, \mathcal{L}_{nn}(\mathsf{D},g(f(\mathsf{D})+\mathsf{X};\theta^*))\end{aligned} \tag{8}$$

where the dual parameter λ is to be determined via cross validation on the generalization error.

Remark A.1. Optimization problem (8) has three terms. The first term controls the amount of information that leaks to the DNN and we want to minimize this information. The second term controls the amount of uncertainty that we are injecting in the form of independently sampled noise. This term is typically proportional to the variance of the noise and we want this term to be maximized (and hence the negative sign). The last term controls the amount of degradation in the performance of the DNN and we want to minimize that with respect to the knob λ.

Remark A.2. The loss function $\mathcal{L}_{nn}(\cdot)$ is determined at the training time of the DNN to obtain the optimal parameter θ^*. The loss function for a regression problem can be a Lebesgue L_p-norm and for a q-class classification problem with $\mathsf{Z} = g(f(\mathsf{D}) + \mathsf{X};\theta^*)$, it can be $\mathcal{L}_{nn}(\mathsf{D},\mathsf{Z}) = -\sum_{j=1}^{q} \mathbf{1}_{\mathsf{X}\in\mathcal{C}_j} \log(\mathsf{Z}_j)$ where $\mathbf{1}_{\cdot}$ is the indicator function, $\mathcal{C}_j$ represents the j^{th} class and Z_j is the j^{th} logit representing the probability that $\mathsf{D}\in\mathcal{C}_j$. Suppose $\mathbf{1}_{\mathsf{D}}$ is a one-hot-encoded q-vector with j^{th} element being 1 if $\mathsf{D}\in\mathcal{C}_j$ and the rest are zero. We then can write the classification loss in vector form as $\mathcal{L}_{nn}(\mathsf{D},\mathsf{Z}) = -\mathbf{1}_{\mathsf{X}}^T \log(\mathsf{Z})$. For the remainder of this paper, we target a q-class classification and rewrite (8) as

$$\begin{aligned}\mathsf{X}^* = \operatorname{argmin}_{\mathsf{X}} \; &\mathcal{H}(f(\mathsf{D})+\mathsf{X}) - \mathcal{H}(\mathsf{X}) \\ &+ \lambda\, \mathbf{1}_{\mathsf{X}}^T \log\big(g(f(\mathsf{D})+\mathsf{X};\theta^*)\big)\end{aligned} \tag{9}$$

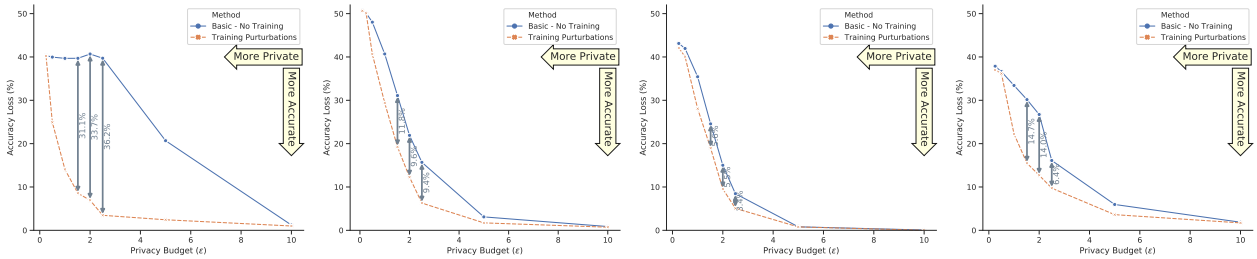
Remark A.3. Assuming X is element-wise distributed as $Lap(\mathsf{M},\mathsf{B})$, we can establish that the first and second term in (9) are proportional to $\log(\mathsf{B})$ and rewrite the optimization problem as

$$\begin{aligned}(\mathsf{M}^*,\mathsf{B}^*) = \operatorname{argmin}_{\mathsf{M},\mathsf{B}} \; &-\log(\mathsf{B}) \\ &+ \lambda\, \mathbb{E}_{\mathsf{X}\sim Lap(\mathsf{M},\mathsf{B})}\big[\mathbf{1}_{\mathsf{X}}^T \log\big(g(f(\mathsf{D})+\mathsf{X};\theta^*)\big)\big]\end{aligned} \tag{10}$$

where $\mathbb{E}[\cdot]$ represents the expectation that can be approximated via Monte Carlo sampling of X. Solving this optimization problem, we get the optimal noise distribution parameters.

Inference: In order to do the inference, we generate a noise tensor X drawn from the distribution we learned in (10), and add it to the output of function f before sending it to the server.

Guarantee: The optimization problem (10) guarantees that for a given value of the knob λ, one can find the distribution



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

Figure 5. Accuracy loss vs. the privacy budget (ϵ) for the four benchmark datasets, measured on test set, for privacy budget(ϵ of up to 10). *Basic -No Training* represents the results of adding random differentially-private perturbation. *Training Purturbations* shows the effect of training the perturbation distribution parameters. The rest of the neural network remains frozen at all times. The gaps shown with vertical arrows indicate the improvement in accuracy brought by Cloak's perturbation training for a given ϵ.

Table 1. Neural networks used in experiments of Section 4.

Neural Network	Conv Layers	FC Layers	Dataset
VGG-16	13	3	CelebA
AlexNet	5	3	CIFAR-100
LeNet-5	13	3	MNIST
VGG-16 (modified)	3	2	UTKFace

parameters that minimizes the information sent to the cloud, while preserving the quality of the service. Since this optimization problem is convex, we are guaranteed to converge to the optimal point via gradient descent algorithm. That is why Cloak uses this optimization technique.

A.2. Detailed Experimental Setup

A.2.1. Experimentation Hardware and OS

We have run the experiments for CelebA and CIFAR-100 datasets on an Nvidia RTX 2080 Ti GPU, with 11GB VRAM, paired with 10 Intel Core i9-9820X processors with 64GBs of memory. The experiments for MNIST and UTKFace datasets were run on the CPU. The systems runs an Ubuntu 18.04 OS, with CUDA version V10.2.89.

A.2.2. Neural Network Architectures

Table 1 shows the DNNs used for the experiments. The code for all the models is available in the supplementary materials. The modified VGG-16 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).

A.2.3. Mutual Information Estimation

As mentioned in Section 4, the mutual information between the input images and their noisy representations are estimated over the test set images using ITE (Szabó, 2014) toolbox's Shannon mutual information estimator. For MNIST images, our dataset had inputs of size 32×32 pixels, which we flattened to 1024 element vectors, for estimating the mutual information. For other datasets, since the images were larger ($32 \times 32 \times 3$), there were more dimensions and mutual information estimation was not accurate. So, what we did here was calculate mutual information channel by channel (i.e. we estimated the mutual information between the red channel of the image and its noisy representation, then the green channel and then blue), and we averaged over all channels.

The numbers reported in 4.2 are remnant mutual information percentages, which means the remaining mutual information is divided by the information content in the original images. This information content was estimated using self-information (Shannon information), using the same toolbox.

A.2.4. Hyperparameters for Training

Tables 2, 3 and 4 show the hyperparameters used for training in the experiments of Sections 4.1, 4.2 and 4.3, respectively. For the last two, 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 experiment in Figure 1 are same as the ones for experiment in Section 4.3, which is in Table 4. In our implementation, for ease of use and without loss of generality, we have introduced a variable γ to the loss function in Equation 4, in a way that $\gamma = \frac{1}{\lambda}$, and it is the coefficient of the mutual information term, not the cross-entropy loss. 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) and we need not set λ very high when we want to remove the mutual information term. We just set γ to zero. 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 mutual information term was not used. 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 for CelebA. For testing, as mentioned in the text of Section 4, 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, as mentioned in Section 3.4, since we do mini-batch training and for each

mini-batch we take a new sample. Finally, M_x is set to 2.0 for all benchmarks, except for MNIST where it is set to be 1.5.

A.3. Code Directory Structure

The `Code.zip` is available in the supplementary material. Also, the saved models and numpy files are available at the link `http://doi.org/10.5281/zenodo.3665794`. Both the `Code.zip` and the `saved_models_npys.zip` which is provided at the link have the same structure. They each contain 3 Folders named *Experiment1, Experiment2* and *Experiment3-celeba* which are related to Sections 4.1, 4.2 and 4.3, respectively. Within each experiment folder, there are 4 folders that correspond to the 4 datasets and the code is provided for each dataset in the related folder. Also, the pre-trained parameters needed are provided in the `saved_models_npys.zip`, in the corresponding directory. So, all that is needed to be done is to copy all files from the `saved_models_npys.zip` directory to their corresponding positions in the Code folder, and then run the provided Jupyter notebooks. For CelebA dataset, as mentioned, the experiment in Section 4.2 is carried out by visualizations. Therefore, the visualization notebook is provided with some sample images saved in it. For the other datasets, however, the noisy and original representations are provided (in .npy format), along with the script used to estimate mutual information. Also, the notebook that was used to generate those representations is provided, in case someone wants to reproduce the results, and the saved (noise-trained) 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 within the Experiment1 and 2 folders that has train 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.2.4. Finally, you can run a test function that is provided to evaluate the model. For Experiment2, at the end of the training notebooks there is also an script that generates the original and noisy representations for mutual information estimation. 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 before hand, if you place the provided `.npy` files in the correct directories. For the mutual information estimation you will need to download the ITE toolbox (Szabó, 2014). The link is provided in the code.

A.4. Section 4.1 results with broader ϵ range

Figure 5 re-illustates Figure 2 in Section 4.1, with a wider privacy budget of up to 10. It can be seen that for most benchmarks, for privacy budgets larger than 5, since the amount of injected noise is less, the accuracy loss is less and the training the perturbation parameters makes a more moderate improvement.

Table 2. Hyperparameters for Section 4.1 experiments.

Dataset	Epsilon	Details
CelebA	10.00	0.3 epoch w/ lr=0.0001
	5.00	0.3 epoch w/ lr=0.01, 0.3 epoch w/ lr=0.0001
	2.50	0.3 epoch w/ lr=0.01, 0.3 epoch w/ lr=0.0001
	2.00	1 epoch w/ lr=0.01, 1 epoch w/ lr=0.00001
	1.52	1 epoch w/ lr=0.01, 1 epoch w/ lr=0.00001
	1.00	1 epoch w/ lr=0.01, 0.5 epoch w/ lr=0.00001
	0.50	1 epoch w/ lr=0.01, 0.5 epoch w/ lr=0.00001
	0.25	1 epoch w/ lr=0.01, 0.5 epoch w/ lr=0.00001
CIFAR-100	10.00	1 epoch w/ lr= 0.00001
	5.00	1 epoch w/ lr= 0.01, 2 epoch w/ lr=0.001, 3 epoch w/ lr=0.00001
	2.50	9 epoch w/ lr= 0.01, 6 epoch w/ lr= 0.001, 3 epoch w/ lr =0.0001
	2.00	20 epoch w/ lr=0.000001
	1.52	20 epoch w/ lr=0.000001
	1.00	20 epoch w/ lr=0.000001
	0.50	20 epoch w/ lr=0.000001
	0.25	20 epoch w/ lr=0.000001
MNIST	10.00	20 epoch w/ lr=0.001
	5.00	20 epoch w/ lr=0.001
	2.50	30 epoch w/ lr=0.001, 25 w/ lr=0.0001
	2.00	20 epoch w/ lr=0.001, 32 w/ lr=0.0001
	1.52	30 epoch w/ lr=0.001, 25 w/ lr=0.0001
	1.00	30 epoch w/ lr=0.001, 25 w/ lr=0.0001
	0.50	10 epoch w/ lr=0.00001
	0.25	10 epoch w/ lr=0.00001
UTKFace	10.00	1 epoch w/ lr= 0.01, 5 epoch w/ lr=0.0001
	5.00	1 epoch w/ lr= 0.01, 5 epoch w/ lr=0.0001
	2.50	1 epoch w/ lr= 0.01, 8 epoch w/ lr=0.0001
	2.00	6 epoch w/ lr= 0.01, 6 epoch w/ lr=0.0001
	1.52	6 epoch w/ lr= 0.01, 6 epoch w/ lr=0.0001
	1.00	6 epoch w/ lr= 0.01, 6 epoch w/ lr=0.0001
	0.50	12 epoch w/ lr= 0.01, 6 epoch w/ lr=0.0001
	0.25	12 epoch w/ lr= 0.01, 6 epoch w/ lr=0.0001

Table 3. Hyperparameters for Section 4.2 experiments.

Dataset	Point#	Details
CIFAR-100	1	6 epoch w/ lr=0.001 6, 14 epoch w/ lr= 0.00001
	2	10 epoch w/ lr= 0.001 lambda = 1, 2 epoch w/ lr=0.001 lambda =10
	3	10 epoch w/ lr= 0.001 lambda = 1, 2 epoch w/ lr=0.001 lambda = 10
	4	12 epoch w/ lr= 0.001 lambda = 1, 2 epoch w/ lr=0.001 lambda = 10
	5	17 epoch w/ lr= 0.001 lambda = 1, 3 epoch w/ lr=0.001 lambda = 10
	6	24 epoch w/ lr= 0.001 lambda = 1, 2 epoch w/ lr=0.001 lambda = 10
	7	30 epoch w/ lr= 0.001 lambda = 1, 2 epoch w/ lr=0.001 lambda = 10
	8	40 epoch w/ lr= 0.001 lambda = 0.2, 2 epoch w/ lr=0.001 lambda = 10
	9	140 epoch w/ lr= 0.001 lambda = 0.2, 2 epoch w/ lr=0.001 lambda = 10
MNIST	10	300 epoch w/ lr= 0.001 lambda = 0.01, 20 epoch w/ lr=0.0001 lambda = 10
	1	25 epoch w/ lr =0.001, 30 epoch w/ lr=0.0001
	2	50 epoch w/ lr= 0.01 lambda = 100, 40 epoch w/ lr=0.001 lambda = 1000
	3	50 epoch w/ lr= 0.01 lambda = 100, 50 epoch w/ lr=0.001 lambda = 1000
	4	50 epoch w/ lr= 0.01 lambda = 100, 60 epoch w/ lr=0.001 lambda = 200
	5	50 epoch w/ lr= 0.01 lambda = 100, 90 epoch w/ lr=0.001 lambda = 200
	6	50 epoch w/ lr= 0.01 lambda = 100, 160 epoch w/ lr=0.001 lambda = 200
	7	50 epoch w/ lr= 0.01 lambda = 100, 180 epoch w/ lr=0.001 lambda = 200
	8	50 epoch w/ lr= 0.01 lambda = 100, 260 epoch w/ lr=0.001 lambda = 100
	9	50 epoch w/ lr= 0.01 lambda = 100, 290 epoch w/ lr=0.001 lambda = 100
	10	50 epoch w/ lr= 0.01 lambda = 100, 300 epoch w/ lr=0.001 lambda = 100
UTKFace	1	6 epoch w/ lr = 0.01, 6 epoch w/ lr=0.0001
	2	4 epoch w/ lr=0.01 lambda=0.1, 2 epoch w/ lr=0.0001 lambda = 100
	3	8 epoch w/ lr=0.01 lambda=0.1, 2 epoch w/ lr=0.0001 lambda = 100
	4	10 epoch w/ lr=0.01 lambda=0.1, 2 epoch w/ lr=0.0001 lambda = 100
	5	12 epoch w/ lr=0.01 lambda=0.1, 2 epoch w/ lr=0.0001 lambda = 100

Table 4. Hyperparameters for Section 4.3 experiments.

Task	Point#	Details
Gender Classification	1	1 epoch w/ lr=0.0001
	2	1 epoch w/ lr=0.01, 2 epoch w/ lr=0.0001
	3	1 epoch w/ lr=0.01, 2 epoch w/ lr=0.0001, 1 epoch w/ lr=0.00001
	4	1 epoch w/ lr=0.01, 2 epoch w/ lr=0.0001, 1 epoch w/ lr=0.00001
	5	1 epoch w/ lr=0.01, 2 epoch w/ lr=0.0001, 1 epoch w/ lr=0.00001
	6	1 epoch w/ lr=0.01, 2 epoch w/ lr=0.0001, 3 epoch w/ lr=0.00001
	7	1 epoch w/ lr=0.01, 2 epoch w/ lr=0.0001, 2 epoch w/ lr=0.00001
	8	1 epoch w/ lr=0.01, 2 epoch w/ lr=0.0001, 3 epoch w/ lr=0.00001
Smile Detection	1	1 epoch w/ lr=0.01, 0.1 epoch w/ lr=0.001
	2	0.5 epoch w/ lr=0.01 lambda=1, 0.2 epoch w/ lr=0.001 lambda=1
	3	0.5 epoch w/ lr=0.01 lambda=1, 0.4 epoch w/ lr=0.001 lambda=1
	4	0.5 epoch w/ lr=0.01 lambda=1, 0.5 epoch w/ lr=0.001 lambda=1
	5	0.5 epoch w/ lr=0.01 lambda=1, 0.7 epoch w/ lr=0.001 lambda=1
	6	0.5 epoch w/ lr=0.01 lambda=1, 0.8 epoch w/ lr=0.001 lambda=1
	7	0.5 epoch w/ lr=0.01 lambda=1, 0.8 epoch w/ lr=0.001 lambda=1, 0.2 epoch w/ lr=0.001 lambda=5
	8	0.5 epoch w/ lr=0.01 lambda=1, 0.8 epoch w/ lr=0.001 lambda=1, 0.2 epoch w/ lr=0.001 lambda=100