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]
[1] University of California San Diego, [2] Amazon
{fmireshg, mtaram, a1yousse, tullsen, hadi}@eng.ucsd.com
ajjalali@amazon.com

Abstract

This work shows that, in many cases, only a small portion of the input is required for a cloud-based machine learning service to offer an accurate prediction. Discovering this subset is one of the main objectives of this paper. We formulate this problem as a gradient-based perturbation maximization method that discovers this subset in the input feature space with respect to the decision making of the prediction model used by the provider. After identifying the essential subset, our framework, Cloak, suppresses the rest of the features in the consumer's input and only sends the essential ones to the cloud. As such, the service provider can use those features to return an accurate prediction and also to improve its service, while at the same time the privacy of the consumer is better protected. We also demonstrate in our experiments that by removing the extra features, the post-hoc fairness of the classifier is improved as well.

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 betterment of its services [1], 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 [2, 3, 4, 5]. 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 $318\times$ to $14{,}000\times$ 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 [6, 7]. 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.

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

Cryptographic Technique	Release Year	DNN	Dataset	Prediction Time (sec) Encry.	Conv.	Cloak	Slowdown
FALCON [8]	2020	VGG-16	ImageNet	12.96	0.0145	0.0148	906×
DELPHI [9]	2020	ResNet-32	CIFAR-100	3.5	0.0112	0.0113	318×
CrypTen [10]	2019	ResNet-18	ImageNet	8.30	0.0121	0.0123	691×
GAZELLE [11]	2018	ResNet-32	CIFAR-100	82.00	0.0112	0.0113	7,454×
MiniONN [12]	2017	LeNet-5	MNIST	9.32	0.0007	0.0007	14,121×

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*.

Preprint. Under review.

The intuition is that if a feature can consistently tolerate 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 known analytical function. The larger the scales, the larger the noise that can be added to a corresponding feature, and the less conducive the features 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 shows that Cloak reduces the MI between input images and the publicized representation by 85.01% with accuracy loss of only 1.42%.

Optimization Problem: 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 $\mathbf{r} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma}$ is a diagonal covariance matrix, as we set the elements of the noise to be independent. This noisy representation helps find the conducive features. 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:

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

To solve this, we bound the terms of this equation, and then take an iterative approach [13]. we find an upper bound for $I(\mathbf{x_c}; \mathbf{u})$ and a lower bound for $I(\mathbf{x_c}; \mathbf{c})$ (appendix A.1).

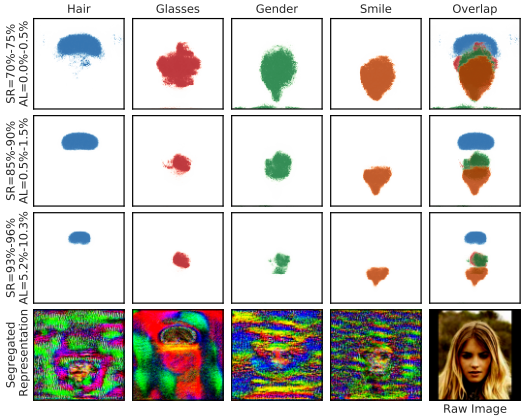


Figure 1: Cloak's discovered features for target DNN classifiers (VGG-16) for black-hair color, eyeglasses, gender, and smile detection. The colored features are conducive to the task. The 3 sets of features depicted for each task correspond to different suppression ratios (SR). AL denotes the range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

2 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 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.

Noise Re-parameterization. To solve the optimization problem above, 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. For this purpose, we applied some constraints on the noise distribution standard deviation and re-parameterized it. The details are in the appendix A.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 std substitute trainable tensor ($\boldsymbol{\rho}$, where $\boldsymbol{\sigma} = (1.0 + \tanh(\boldsymbol{\rho}))/2.(M)$) with a large negative number. This generates the initial value of zero for the standard deviations. In optimization step, the loss is computed on a batch of training data and the gradient of the loss with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ are calculated. Since the loss (appendix, Equation 8) incorporates expected value over noise samples, Cloak uses Monte Carlo sampling [14] with sufficiently large number of noise samples to calculate the loss. Once the training is finished, the optimized mean and std tensors are collected and passed to the next phase.

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, 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).

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 black pixels might not match the distribution of the data that the classifier expects. To address this, we find a suppressed representation (Section A.1.4), 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 cross-entropy loss as our loss function.

Algorithm 1 Perturbation Training

Input: $\mathcal{D}, y, f_\theta, m, \lambda$
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. (8)
until Algorithm converges
Return: $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$

Algorithm 2 Suppression-Value Training

Input: $\mathcal{D}, y, f_\theta, \boldsymbol{\sigma}, \boldsymbol{\mu}, \mathbf{b}$
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: $\boldsymbol{\mu}_s$

Online Prediction. Cloak's online prediction is computationally efficient; it only adds noise sampling, masking, and addition to the conventional prediction. First a noise tensor sampled from the optimized distribution $\mathcal{N}(0,\boldsymbol{\sigma}^2)$ is added to the input, then the binary mask b is applied to the noisy input image. Finally $\boldsymbol{\mu}_s$ 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 using the noise maps from the third row. The non-conducive features are removed and replaced with $\boldsymbol{\mu}_s$. The conducive features, however, are visible.

3 Experimental Results

To evaluate Cloak, we use four real-world datasets on four Deep Neural Networks (DNNs). Details are provided in A.4.

Privacy-Accuracy Trade-Off. Figure 2a shows accuracy loss of the DNN classifier vs. the loss in mutual information using sifted representations. 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. 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 UTKFace, Cloak reduces the information in the input significantly (93% and 85% respectively) with little loss in accuracy (0.5% and 2.7%). 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. Results for other DNNs and datasets can be found in the appendix (Fig. 4).

Adversary to Infer Information. 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. First, the adversary who has access to an 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 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: eyeglasses and

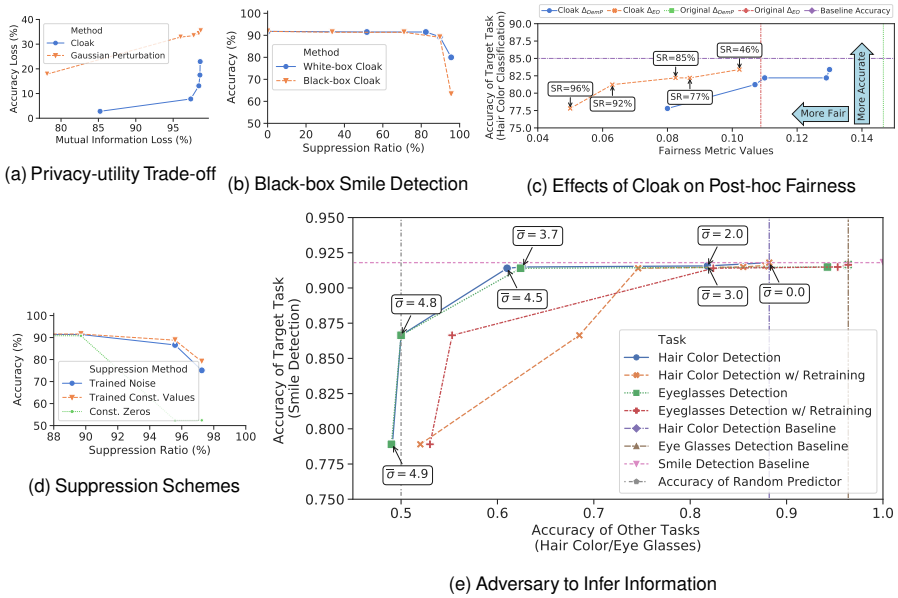


Figure 2: (a) Privacy-utility trade-off for UTKFace.(b) performance of Cloak in a black-box setting. (c) Post-hoc fairness. (d) compares different suppression methods. (e) Protection from an adversary that tires to infer black-hair color or eyeglasses from the sifted representations.

hair color. The adversaries have pre-trained classifiers for both these tasks. Figure 2e shows the results of this experiment. Each point in this figure is generated using a noise map with a Suppression Ratio (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%), they still end-up with near random classifiers, 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 hair than with the eyeglasses.

Black-Box Access Mode. We show that it is possible for Cloak to protect users' privacy even with 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 or its parameters and is only allowed to send requests and get back responses. In this setting, following similar methodology to the methodology described in Shokri et al. [15] we train a substitute model that helps us train Cloak's representations. Since we assume no knowledge of the model architecture, Cloak substitutes the target classifiers with another architecture, i.e, ResNet18 with VGG-16. After training the substitute model, we apply Cloak to them to find noise maps and suppressed representations. Figure 2b and 3 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 [16, 17].

Post-hoc Effects of on Fairness. Cloak, by removing extra features, not only benefits privacy but can also remove unintended biases of the classifier. In many cases the features that bias the classifiers highly overlap with the non-conducive features that Cloak discovers. Here we evaluates this positive side-effect of Cloak by adopting a setup similar to that of Kairouz et al. [18]. 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 measure two metrics: the difference in Demographic Parity (Δ_{DemP}), and the difference in Equal Opportunity (Δ_{EO}). Figure 2c shows that as Cloak suppresses more non-conducive features, the fairness metrics improve significantly. 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.

Different suppression schemes. Figure 2d shows the accuracy of the suppression schemes described in Section 2 on the smile detection task (on CelebA w/ VGG-16). Among different schemes, suppression using the trained values yields the best accuracy.

4 Conclusion

We propose Cloak, a mechanism that finds features in the data that are unimportant for a cloud ML prediction model. That enables Cloak to suppress those features before publicizing them, providing only the minimum information exposure. In doing so, Cloak not only minimizes the impact on the utility of the service, it also imposes minimal overhead on the response time of the prediction service.

References

[1] O. Rana and J. Weinman, "Data as a currency and cloud-based data lockers," *IEEE Cloud Computing*, vol. 2, no. 2, pp. 16–20, 2015.

[2] P. Kocher, J. Horn, A. Fogh, , D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *IEEE Symposium on Security and Privacy (S&P)*, 2019.

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

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

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

[6] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, *et al.*, "MLPerf inference benchmark," in *International Symposium on Computer Architecture (ISCA)*, 2020.

[7] MLPerf Organization, "MLPerf Benchmark Suite," 2020. url: *https://mlperf.org*.

[8] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: Honest-majority maliciously secure framework for private deep learning," *arXiv preprint arXiv:2004.02229*, 2020.

[9] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *USENIX Security Symposium (USENIX Security)*, 2020.

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

[11] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *USENIX Security Symposium (USENIX Security)*, 2018.

[12] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[13] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

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

[15] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[16] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ACM on Asia conference on computer and communications security (AsiaCCS)*, 2017.

[17] D. Arpit, S. Jastrzkebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, *et al.*, "A closer look at memorization in deep networks," in *International Conference on Machine Learning (ICML)*, 2017.

[18] P. Kairouz, J. Liao, C. Huang, and L. Sankar, "Censored and fair universal representations using generative adversarial models," *arXiv preprint arXiv:1910.00411*, 2019.

[19] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[20] N. J. Beaudry and R. Renner, "An intuitive proof of the data processing inequality," *arXiv preprint arXiv:1107.0740*, 2011.

[21] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *arXiv preprint arXiv:0912.0071*, 2009.

[22] K. Chaudhuri, A. D. Sarwate, and K. Sinha, "A near-optimal algorithm for differentially-private principal components," *J. Mach. Learn. Res.*, vol. 14, p. 2905–2943, Jan. 2013.

[23] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, pp. 211–407, Aug. 2014.

[24] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[25] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *ACM Conference on Computer and Communications Security (CCS)*, 2015.

[26] N. Papernot, M. Abadi, Úlfar Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," *arXiv preprint arXiv:1610.05755*, 2016.

[27] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with pate," *arXiv preprint arXiv:1802.08908*, 2018.

[28] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv:1711.05189*, 2017.

[29] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[30] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning (ICML)*, 2016.

[31] N. Agrawal, A. Shahin Shamsabadi, M. J. Kusner, and A. Gascón, "Quotient: two-party secure neural network training and prediction," in *ACM Conference on Computer and Communications Security (CCS)*, 2019.

[32] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *International Conference on Learning Representations (ICLR)*, 2019.

[33] L. Hanzlik, Y. Zhang, K. Grosse, A. Salem, M. Augustin, M. Backes, and M. Fritz, "Mlcapsule: Guarded offline deployment of machine learning as a service," *arXiv preprint arXiv:1808.00590*, 2018.

[34] K. G. Narra, Z. Lin, Y. Wang, K. Balasubramaniam, and M. Annavaram, "Privacy-preserving inference in machine learning services using trusted execution environments," *arXiv preprint arXiv:1912.03485*, 2019.

[35] S. A. Osia, A. S. Shamsabadi, S. Sajadmanesh, A. Taheri, K. Katevas, H. R. Rabiee, N. D. Lane, and H. Haddadi, "A hybrid deep learning architecture for privacy-preserving mobile analytics," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[36] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy," *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[37] F. Mireshghallah, M. Taram, P. Ramrakhyani, A. Jalali, D. Tullsen, and H. Esmaeilzadeh, "Shredder: Learning noise distributions to protect inference privacy," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[38] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *International Conference on Computer Vision (ICCV)*, 2015.

[39] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-100 (canadian institute for advanced research)," url: *http://www.cs.toronto.edu/~kriz/cifar.html*.

[40] Z. Zhang, Y. Song, and H. Qi, "Age progression/regression by conditional adversarial autoencoder," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4352–4360, 2017.

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

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, 2012.

[45] Y. LeCun, "Gradient-based learning applied to document recognition," 1998.

[46] Z. Szabó, "Information theoretical estimators toolbox," *Journal of Machine Learning Research*, vol. 15, pp. 283–287, 2014.

[47] D. Madras, E. Creager, T. Pitassi, and R. Zemel, "Learning adversarially fair and transferable representations," *arXiv preprint arXiv:1802.06309*, 2018.

A Appendix

A.1 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 $\mathbf{r} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma}$ 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:

$$\min_{\mathbf{x_c}} \quad I(\mathbf{x_c}; \mathbf{u}) - \lambda I(\mathbf{x_c}; \mathbf{c}) \tag{2}$$

To solve this problem, we bound the terms of our optimization problem of Equation 2, and then take an iterative approach [13]. we find an upper bound for $I(\mathbf{x_c}; \mathbf{u})$ and a lower bound for $I(\mathbf{x_c}; \mathbf{c})$.

A.1.1 Upper bound on $I(\mathbf{x_c};\mathbf{u})$

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

$$I(\mathbf{x_c}; \mathbf{u}) \leq I(\mathbf{x_c}; \mathbf{x}) = \mathcal{H}(\mathbf{x_c}) - \frac{1}{2}\log((2\pi e)^n |\boldsymbol{\Sigma}|) \tag{3}$$

Where $\mathcal{H}(\mathbf{x_c}|\mathbf{x})$ is the entropy of the added Gaussian noise. Here $|\boldsymbol{\Sigma}|$ denotes the determinant of the covariance matrix. Then by applying Theorem 8.6.5 from [19] which gives an upper bound for the entropy, to $\mathbf{x_c}$, we can write:

$$I(\mathbf{x_c}; \mathbf{u}) \leq \frac{1}{2}\log((2\pi e)^n \frac{|Cov(\mathbf{x_c})|}{|\boldsymbol{\Sigma}|}) \tag{4}$$

Since $\mathbf{x}$ and $\mathbf{r}$ are independent variables and $\mathbf{x_c} = \mathbf{x} + \mathbf{r}$, we have $|Cov(\mathbf{x_c})| = |Cov(\mathbf{x}) + \boldsymbol{\Sigma}|$. 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 $\boldsymbol{\Sigma}$ is already a diagonal matrix, $|Cov(\mathbf{x}) + \boldsymbol{\Sigma}| = |Q(\Lambda + \boldsymbol{\sigma}^2)Q^T| = \prod_{i=1}^{n}(\lambda_i + \sigma_i^2)$. By substituting this in Equation 4, and simplifying we get the upper bound for $I(\mathbf{x_c}; \mathbf{u})$ as the following:

$$I(\mathbf{x_c}; \mathbf{u}) \leq \frac{1}{2}\log((2\pi e)^n \prod_{i=1}^{n}(1 + \frac{\lambda_i}{\sigma_i^2})) \tag{5}$$

A.1.2 Lower bound on $I(\mathbf{x_c};\mathbf{c})$

Theorem A.1. *The lower bound on $I(\mathbf{x_c};\mathbf{c})$ is:*

$$\mathcal{H}(\mathbf{c}) + \max_{q} \mathbb{E}_{\mathbf{x_c},\mathbf{c}}[\log q(\mathbf{c}|\mathbf{x_c})] \tag{6}$$

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 redacted to save space. □

A.1.3 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 5) and maximize the lower bound (Equation 6):

$$\min_{\boldsymbol{\sigma}, q} \quad \frac{1}{2}\log((2\pi e)^n \prod_{i=1}^{n}(1 + \frac{\lambda_i}{\sigma_i^2})) + \lambda \sum_{\mathbf{c_i}, \mathbf{x_{c_i}}} (-\log q(\mathbf{c_i}|\mathbf{x_{c_i}})) \tag{7}$$

We 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. 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:

$$\mathcal{L} = -\log \frac{1}{n}\sum_{i=0}^{n}\sigma_i^2 + \lambda \mathbb{E}_{\mathbf{r} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma^2}), \mathbf{x} \sim \mathcal{D}} \Big[-\sum_{k=1}^{K} y_k \log(f_\theta(\mathbf{x}+\mathbf{r}))_k \Big] \tag{8}$$

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 decrease the classification error and maintains accuracy. The parameter λ is a knob which provides a trade-off between these two.

A.1.4 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 [20], 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 6 also holds for $I(\mathbf{x_s}; \mathbf{c})$.

A.2 Noise Re-parameterization

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.

A.3 Related Work

For *training*, the literature abounds with the studies that use noise addition as a randomization mechanism to protect privacy [21, 22, 23, 24, 25, 26, 27]. Many of these studies have applied differential privacy to a training setting where they concern with leaking private information in training set through the machine learning model. Privacy on offloaded computation can also be provided by the means of cryptographic tools such as homomorphic encryption and/or Secure Multiparty Computation (SMC) [28, 11, 29, 30, 12, 9, 8, 31]. However, these approaches suffer from a prohibitive computational costs (See Table 1), on both cloud and user side, exacerbating the complexity and compute-intensivity of neural networks especially on resource-constrained edge devices. Several other research [32, 33, 34] rely on trusted execution environments to remotely run ML algorithm. 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 [2, 4].

Only a handful of studies have addressed privacy of prediction by adding noise to the data. Osia et al. [35] employed dimensionality reduction techniques to reduce the amount of information before sending to untrusted cloud. Wang et al. [36] propose a noise injection framework that randomly nullifies input elements for private inference, but their method requires retraining of the entire network. In a more recent work [37], Mireshghallah et al. propose 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). Due to the heuristic and pattern-based nature of this prior work, it does not provide formal guarantees. In contrast, Cloak's approach is to directly learn conducive features and suppress non-conducive ones with learned constant values. This enables us to theoretically show we decrease the upper bound on the mutual information between the input and publicized representation, and increase the lower bound on utility. 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. Therefore, it requires collaboration of the service provider and change in the infrastructure. In contrast, we show that Cloak can perform equally efficient in black-box settings without collaboration of the service provider.

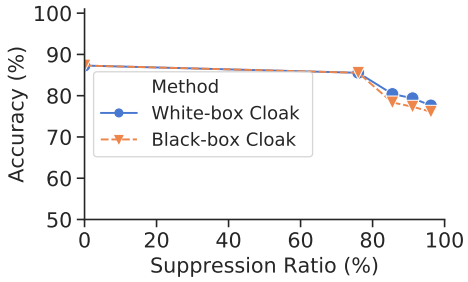
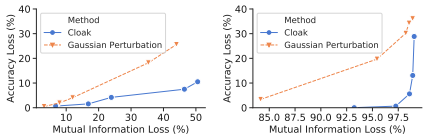


Figure 3: Black-box hair color



(a) CIFAR-100 (b) MNIST

Figure 4: Privacy-Utility Trade-off for More Datasets

A.4 Detailed Experimental Setup

There are four datasets used in our evaluations: CelebA [38], CIFAR-100 [39], UTKFace [40] and MNIST [41]. We have used these datasets with VGG-16 [42], ResNet-18 [43], AlexNet [44], VGG-16 (modified), and LeNet-5 [45] 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 $224 \times 224 \times 3$, $32 \times 32 \times 3$, $32 \times 32 \times 3$, 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. 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 [46].

A.4.1 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 systems runs an Ubuntu 18.04 OS, with CUDA version V10.2.89.

A.4.2 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%.

A.4.3 Mutual Information Estimation

The mutual information between the input images and their noisy representations are estimated over the test set images using ITE [46] 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.

Table 2: hyper parameters for Section 3

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 3

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

The numbers reported in Section 3 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.

A.4.4 Hyperparameters for Training

Tables 2, 3 and 4 show the hyperparameters used for training. 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 8, 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.

Table 4: hyper parameters for Section 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 [47].