Shredder: Learning Noise to Protect Privacy with Partial DNN Inference on the Edge

Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Dean Tullsen, Hadi Esmaeilzadeh
fmireshg, mtaram @eng.ucsd.edu
prakash.ramrakhyani@arm.com
tullsen, hadi @eng.ucsd.edu

Abstract

A wide variety of DNN applications increasingly rely on the cloud to perform their huge computation. This heavy trend toward cloud-hosted inference services raises serious privacy concerns. This model requires the sending of private and privileged data over the network to remote servers, exposing it to the service provider. Even if the provider is trusted, the data can still be vulnerable over communication channels or via side-channel attacks [1, 2] at the provider. To that end, this paper aims to reduce the information content of the communicated data without compromising the cloud service's ability to provide a DNN inference with acceptably high accuracy. This paper presents an end-to-end framework, called Shredder, that, without altering the topology or the weights of a pre-trained network, learns an additive noise distribution that significantly reduces the information content of communicated data while maintaining the inference accuracy. Shredder learns the additive noise by casting it as a tensor of trainable parameters enabling us to devise a loss functions that strikes a balance between accuracy and information degradation. The loss function exposes a knob for a disciplined and controlled asymmetric trade-off between privacy and accuracy. While keeping the DNN intact, Shredder enables inference on noisy data without the need to update the model or the cloud. Experimentation with real-world DNNs shows that Shredder reduces the mutual information between the input and the communicated data to the cloud by 70.2% compared to the original execution while only sacrificing 1.46% loss in accuracy.

1 Introduction

Online services that utilize the cloud infrastructure are now ubiquitous and dominate the IT industry [3, 4, 5]. The limited computation capability of edge devices [6] and the increasing processing demand of learning models [7, 8] has naturally pushed most of the computation to the cloud [9]. Coupled with the advances in learning and deep learning, this shift has also enabled online services to offer a more personalized and more natural interface to the users [10]. These services continuously receive raw, and in many cases, personal data that needs to be stored, parsed, and turned into insights and actions. In many cases, such as home automation or personal assistant, there is a rather continuous flow of personal data to the service providers for real-time inference. While this model of cloud computing has enabled unprecedented capabilities due to the sheer power of remote warehouse-scale data processing, it can significantly compromise user privacy. When data is processed on the service provider cloud, it can be compromised through side-channel hardware attacks (e.g., Spectre [1] or Meltdown [2]) or deficiency in the software stack [11]. But even in the absence of such attacks, the service provider can share the data with business partners [12] or government agencies [13]. Although the industry has adopted privacy techniques for data collection and model training [14, 15], scant attention has been given to the privacy of users who increasingly rely on online services for inference.

As Figure 1 illustrates, researchers have attempted to grapple with this problem by employing cryptographic techniques such as multiparty execution [16, 17] and homomorphic encryption [18, 19, 20, 21] in the context of DNNs. However, these approaches suffer from a prohibitive computational and communication cost, exacerbating the already complex and compute-intensive neural network models. Worse still, this burdens additional encryption and decryption layers to the already constrained edge devices despite the computational limit being the main incentive of offloading the inference to the cloud.

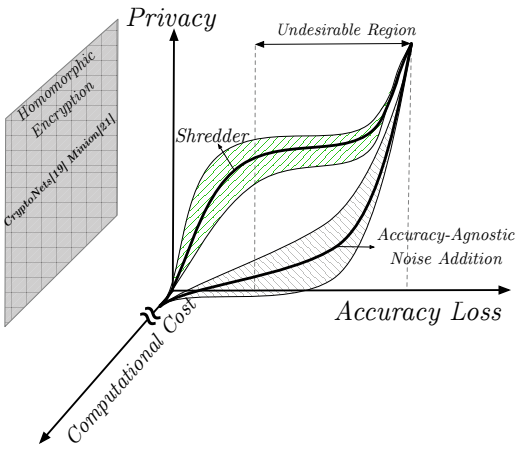


Figure 1: The landscape of research in inference privacy and how Shredder fits in the picture.

This paper, as depicted in Figure 1, takes an orthogonal approach, dubbed Shredder, and aims to reduce the information content of the remotely communicated data through noise injection without imposing significant computational cost. However, as shown, noise injection can lead to a significant loss in accuracy if not administered with care and discipline. To address this challenge, Shredder learns a noise tensor with respect to a loss function that incorporates both accuracy and a measure of privacy. Learning noise does not require retraining the network weight parameters or changing its topological architectures. This rather non-intrusive approach is particularly interesting as most enterprise DNN models are proprietary, and retraining hundred of millions of parameters is resource and time consuming. Shredder, in contrast, learns a much smaller noise tensor through a gradient-driven optimization process.

The main insight is that the noise can be seen as an added trainable set of parameters that can be discovered through an end-to-end training algorithm. We have devised the noise training loss such that it exposes an asymmetric tradeoff between accuracy and privacy as depicted in Figure 1. As such the same model can be run on the same cloud on intentionally noisy data without the need for retraining or the added cost of supporting cryptographic computation. The objective is to minimize the accuracy loss while maximally reducing the information content of the data that a user sends to cloud for an inference service. This problem of offloaded inference is different than the classical differential privacy [22] setting where the main concern is the amount of indistinguishability of an algorithm, i.e., how the output of the algorithm changes if a single user opts out of the input set. In inference privacy, however, the issue is the amount of raw information that is sent out. As such, Shanon's Mutual Information (MI) [23] between the user's raw input and the communicated data to the cloud is used as a measure to quantitively discuss privacy.

Empirical analysis shows Shredder reduces the mutual information between the input and the communicated data by 70.2% compared to the original execution with only 1.46% accuracy loss. With these encouraging results the paper marks an initial step in casting noise-injection to protect privacy as finding a tensor of trainable parameters through an optimization process, doing so without retraining the network weights, and incorporating both privacy and accuracy in the optimization loss. You can find Shredder's code at `https://github.com/shreddercode/Shredder.git`.

2 Shredder: Noise Learning Framework

This section delves deeper into the details of Shredder, starting from describing the problem formulation and how the trainable noise tensor fits into the context. In addition this section describes the loss function and training process that Shredder uses for finding the desired noise tensor. We also describe our privacy model and the notions of privacy that we use.

2.1 Trainable Noise Tensor

Given a pre-trained network $f(x, \theta)$ with K layers and pretrained parameters θ, we choose a cutting point, $layer_c$, where the computation of all the layers $[0..layer_c]$ are made on the edge. We call this the local network, $L(x, \theta_1)$, where θ_1 is a subset of θ from the original model.

The remaining layers, i.e., $[(layer_c + 1)..layer_{K-1}]$, are deployed on the cloud. We call this remote network, $R(x, \theta_2)$. This is shown in Figure 2.

The user provides input x to the local network, and an intermediate activation tensor $a = L(x, \theta_1)$ is produced. Then, a noise tensor n is added to the output of the first part, $a' = a + n$. This a' is then communicated to the cloud where $R(a', \theta_2)$ is computed on noisy data and produces the result $y = f'(x, n, \theta)$ that is sent back to the user.

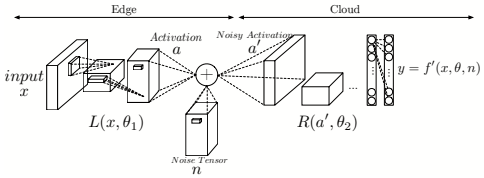


Figure 2: Noise injection in Shredder.

The objective is to find the noise tensor n that minimizes our loss function (Section 2.4). To be able to do this through a gradient based method of optimization, we must find the $\partial y / \partial n$:

$$\frac{\partial y}{\partial n} = \frac{\partial f'(x, \theta, n)}{\partial n} = \frac{\partial R((a+n), \theta_2)}{\partial n} = \frac{\partial layer_{K-1}}{\partial Layer_{K-2}} \times ... \times \frac{\partial Layer_{c+1}}{\partial (a+n)} \times \underbrace{\frac{\partial (a+n)}{\partial n}}_{=\frac{\partial n}{\partial n}=1}$$

Since $L(x, \theta_1)$ is not a function of n, it is not involved in the backpropagation. Gradient of R is also computed through chain rule as shown above. Therefore, the output is differentiable with respect to the noise tensor.

2.2 Ex Vivo Notion of Privacy

To measure the privacy, we look at how much information is leaked from input of the network to the data sent across to the cloud. We define information leakage as the mutual information between x and a, i.e., $I(x, a)$, where

$$I(x; a) = \int_x \int_a p_{x,a} log_2 \frac{p_{x,a}}{p_x p_a} dx da. \tag{1}$$

Mutual information has been widely used in the literature for both understanding the behaviour of neural networks [24, 25, 24, 26, 27, 28], and also to quantify information leakage in anonymity systems in the context of databases [29, 30, 31]. We also use the reverse of mutual information ($1/MI$) as our main and final notion of privacy and call it ex vivo privacy. In our setting, we quantify the information between the user-provided input and the intermediate state that is sent to the cloud. Note that the mutual information is considered an information theoretic notion, therefore it quantifies the average amount of information about the input (x) that is contained in the intermediate state (a). For example, if x and a become independent, $I(x, a) = 0$, and if $a = x$, then the mutual information becomes the maximum value of $I(x, a) = H(x)$, where $H(x)$ is Shanon's entropy of the random variable x.

2.3 In Vivo Notion of Privacy

As the final goal, Shredder reduces the mutual information between x and a'; however, calculating the mutual information at every step of the training is too computationally intensive. Therefore, instead we introduce an in vivo notion of privacy whose whole purpose is to guide our noise training process towards better privacy, i.e, higher $1/MI$. To this end, we use the reverse of signal to noise ratio ($1/SNR$) as proxy for our ex vivo notion of privacy. Mutual information is shown to be a function of SNR in noisy channels [32, 33]. In addition, in this paper, we empirically investigate the relation between the two and show that SNR is a reasonable choice.

2.4 Loss Function

The objective of the optimization is to find the additive noise distribution in such a way that it minimizes $I(x, a')$ and at the same time maintains the accuracy. In other words, it minimizes $\|R(a, \theta) - R(a', \theta)\|$. Although these two objectives seem to be conflicting, it is still a viable optimization, as the results suggest. The high dimensionality of the activations, their sparsity, and the tolerance of the network to perturbations [34, 35] yields such behavior.

The noise tensor that is added is the same size as the activation it is being added to. The number of elements in this tensor would be the number of trainable parameters in our method. Shredder initializes the noise tensor to a Laplace distribution with location parameter μ and scale parameter b. Similar to the initialization in the traditional networks, our initialization parameters, i.e., b and μ are considered hyperparameters in the training and need to be tuned. This initialization affects the accuracy and amount of noise (privacy) of our model.

We evaluate the privacy of our technique during inference through ex vivo ($1/MI$) notion of privacy. However, during training, calculating MI for each batch update would be extremely compute-intensive. For this reason, Shredder uses an in vivo notion of privacy which uses (SNR) as a proxy to MI [33]. In other words, Shredder incorporates SNR in the loss function to guide the optimization towards increasing privacy. We use the formulation $SNR = E[a^2]/\sigma^2(n)$, where $E[a^2]$ is the expected value of the square of activation tensor, and $\sigma^2(n)$ is the variance of the noise we add. Given the in vivo notion of privacy above, our loss function would be:

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) + \lambda \frac{1}{\sigma^2(n)} \tag{2}$$

Where the first term is cross entropy loss for a classification problem consisting M classes ($y_{o,c}$ indicates whether the observation o belongs to class c and $p_{o,c}$ is the probability given by the network for the observation to belong to class c), and the second term is the inverse of variance of the noise tensor to help it get bigger and thereby, increase in vivo privacy (decrease SNR). λ is a coefficient that controls the impact of in vivo privacy in training. Since the numerator in our SNR formulation is constant, we do not involve it in the calculations. The standard deviation of a group of finite numbers with the range $R = max - min$ is maximized if they are equally divided between the minimum, min, and the maximum, max. This is inline with our observations that show as we push the magnitude of the noise to be bigger, the in vivo privacy would also get bigger. The intuition is that we initialize the noise tensor in a way that some elements are negative and some are positive. The positive ones get bigger, and the negative ones get smaller, therefore, the standard deviation of the noise tensor becomes bigger after each update. That's why we employ a formulation opposite to L2 regularization [36], in order to make the magnitude of noise elements greater. So our loss becomes:

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) - \lambda \sum_{i=1}^{N} |n_i| \tag{3}$$

This applies updates opposite to L2 regularization term (weight decay, and λ is similar to the decay factor), instead of making the noise smaller, it makes its magnitude bigger. The λ exposes a knob here, balancing the accuracy/privacy trade-off. That's why it should be tuned carefully for each network. If it is very big, at each update the noise would get much bigger, impeding the accuracy from improving. And if it is too small, its effect on the noise would be minimal. We use -0.01, -0.001 and -0.0001. In general, as the networks and the number of training parameters get bigger, it is better to make λ smaller to prevent the optimizer from making huge updates and overshooting the accuracy.

When initializing noise with a Laplace distribution, the scale factor of the distribution determines the initial in vivo privacy. Depending on the initial in vivo privacy, initial accuracy and the λ, different scenarios could occur. One scenario is where we tune λ so that the in vivo privacy remains constant, the same as its initial value (within a small fluctuation range) and only the accuracy increases. Another scenario occurs if the initial in vivo privacy is a lot bigger than what is desired (this usually occurs if the initialized noise tensor has a high scale factor) – it is easier (faster in terms of training) to set λ very small or equal to zero and train until accuracy is regained. In this case the in vivo privacy will decrease as the accuracy is increasing, but since it was extremely high before, even after decreasing it is still desirable. One other possibility is that the initial in vivo privacy is lower than what we want and when training starts, it will increase as accuracy increases (or if it is not perturbed much by the initial noise, it stays constant).

2.5 Noise Sampling

We choose Laplace distribution for initialization, and perform training until we reach the desired noise level for the given in vivo privacy ($1/SNR$) and accuracy. At this point we save the noise tensor, and repeat the same process multiple times. This is like sampling from a distribution of noise tensors, all of which yield similar accuracy and noise levels. After enough samples are collected, we have the distribution for the noise tensor. At this point, for each inference, we sample from the distribution, and inject this noise to the activation and send it to the cloud. In this phase we just sample from pre-trained noises and no training takes place here.

3 Empirical Evaluation

This section elaborates our observations in details and brings empirical evidences for the efficacy of Shredder. We discuss the accuracy-privacy trade-off, the noise training process with our loss function, a comparison of the in vivo and ex vivo notions of privacy and finally, a network cutting point trade-off analysis.

Mutual Information (MI) is calculated using the Information Theoretical Estimators Toolbox's [37] Shannon Mutual Information with KL Divergence. In the results reported in upcoming sub-sections, MI is calculated over the shuffled test sets on MNIST [38] dataset for LeNet [39], CIFAR-10 dataset for CIFAR-10 [40], SVHN dataset for SVHN [41], and ImageNet [42] dataset for AlexNet [43]. These photos were shuffled through and chosen at random. Using mutual information as a notion of privacy means that Shredder targets the average case privacy, but does not guarantee the amount of privacy that is offered to each individual user.

Table 1 summarizes our experimental results. It is shown that on the networks, Shredder can achieve on average 70.2% loss in information while inducing 1.46% loss in accuracy. The table also shows that it takes Shredder a short time to train the noise tensor, for instance on AlexNet it is 0.1 epoch.

3.1 Accuracy-Privacy Trade-Off

There is a trade-off between the amount of noise that we incur to the network, and its accuracy. As shown in Figure 1, Shredder attempts to increase privacy while keeping the accuracy intact. Figure 3 shows the level of privacy that can be obtained by losing a given amount of accuracy for LeNet, CIFAR-10, SVHN, and AlexNet. In this Figure, the number of mutual information bits that are lost from the original activation using our method is shown on the Y axis. The cutting point of the networks is their last convolution layer. This can be perceived as the output of the *features* section of the network, if we divide the network into *features* and *classifier* sections.

The *Zero Leakage* line depicts the amount of information that needs to be lost in order to leak no information at all. In other words, this line points to the original number of mutual information bits in the activation that is sent to the cloud, without applying noise. The black dots show the information loss that Shredder provides, given a certain loss in accuracy. These trends are similar to that of Figure 1, since Shredder tries to strip the activation from its excess information, thereby preserving privacy and only keeping the information that is used for the classification task. This is the sharp (high slope) rise in information loss, seen in sub-figures of Figure 3. Once the excess information is gone, what remains is mostly what is needed for inference. That is why there is a point (the low slope horizontal line in the figures) where adding more noise (losing more information bits) causes huge loss in accuracy. The extreme to this case can be seen in 3a, where approaching the *Zero Leakage* line causes about 20% loss in accuracy.

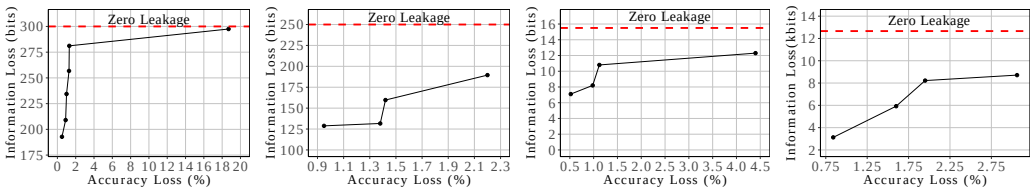
3.2 Loss Function and Noise Training Analysis

As Equation 3 shows, our loss function has an extra term, in comparison to the regular cross entropy loss function. This extra term is intended to help decrease Signal to Noise ratio (SNR). Figure 4 shows part of the training process on AlexNet, cut from its last convolution layer. The black lines show how a regular noise training process would work, with cross entropy loss and Adam Optimizer [44]. As Figure 4a shows in black, the in vivo notion of privacy ($1/SNR$) decreases for regular training as the training moves forward. For Shredder however, the privacy increases and then stabilizes.

This is achieved through tuning of the λ in Equation 3. When the in vivo notion of privacy reaches a certain desired level, λ is decayed to stabilize privacy and facilitate the learning process. If it is not decayed, the privacy will keep increasing and the accuracy would increase more slowly, or even start decreasing. The accuracy, however, increases at a higher pace for regular training, compared to Shredder in Figure 4b. It is noteworthy that this experiment was carried out on the training set of

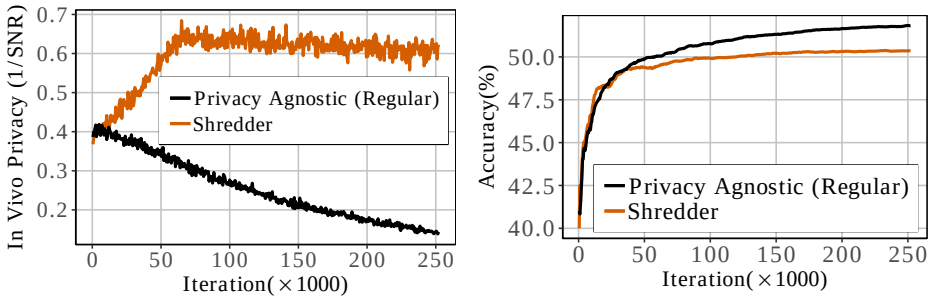
Table 1: Summary of the experimental results of Shredder for the benchmark networks.

Benchmark	LeNet	CIFAR	SVHN	Alexnet	GMean
Original Mutual Information	301.84	236.34	19.2	12661.51	–
Shredded Mutual Information	18.9	90.2	7.1	4439	–
Mutual Information Loss	**93.74%**	**61.83%**	**64.58%**	**64.94%**	**70.2%**
Accuracy Loss	**1.34%**	**1.42%**	**1.12%**	**1.95%**	**1.46%**
Shredder's Learnable Params over Model Size	0.19%	0.65%	0.04%	0.02%	0.10%
Number of Epochs of Training	6.3	1.7	1.2	0.1	1.06



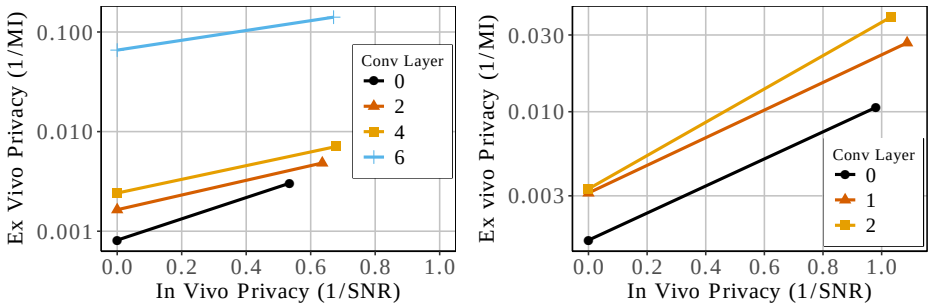
(a) LeNet (b) CIFAR (c) SVHN (d) AlexNet

Figure 3: Accuracy-Privacy trade-off in 4 benchmark networks, cut from their last convolution layer. The zero leakage line shows the original mutual information between input images and activations at the cutting point.



(a) In vivo privacy (b) Accuracy

Figure 4: In vivo notion of privacy and accuracy per iteration of training on AlexNet, when the last convolution layer is the cutting point. The black lines show regular training with cross entropy loss function. The orange lines show Shredder's learning, with loss function shown in Equation 3.



(a) SVHN (b) LeNet

Figure 5: In vivo vs ex vivo notion of privacy in SVHN and LeNet, for different cutting points. The similar slope of the lines show that different layers behave similarly in terms of losing information (ex vivo privacy) when the same amount of noise is applied to them during training (in vivo privacy).

ImageNet, and when the training is finished, there is negligible degradation in accuracy for Shredder on the test set, in comparison to the regularly trained model.

3.3 In Vivo Vs. Ex Vivo Notion of Privacy Analysis

Due to the operations that take place along the execution of different layers of a neural network (e.g. convolution, normalization, pooling, etc.), mutual information between the inputs to the network and the activations keep decreasing as we move forward [27]. So, deeper layers have lower mutual information than the more surface layers, and when noise is injected into them, it is similar to giving the privacy level a head start, since it already has less information compared to a layer on the surface. Figure 5a shows that incurring the same amount of noise (which induces ~ 0.6 in vivo privacy, ~ 1.67 SNR) to convolution layers $0, 2, 4, 6$ of SVHN, causes information loss of $\sim 66\%$ for all four layers. This does not mean the same amountof ex vivo privacy, because each layer had a different starting point for ex vivo privacy. What it means is that the the information loss is proportional to incurred noise and the proportion is consistent over all the 4 layers shown.

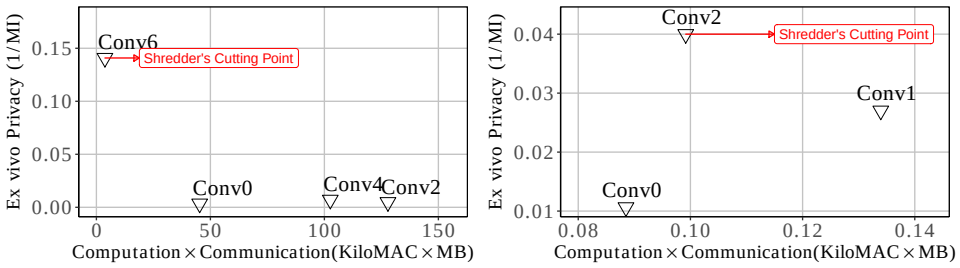


Figure 6: Computation/communication costs and privacy as we select deeper layers in the networks for Shredder's noise addition. Shredder maintains an accuracy loss of $< 2\%$ for all of the cutting points.

3.4 Cutting Point Trade-offs

Layer selection for network cutting point depends on different factors and is mostly an interplay of communication and computation of the edge device. It depends on how many layers of the network the edge device can handle computationally, and how much data it can send through the connection protocols it can support. If we choose deeper layers in the network, we would naturally have a lower mutual information between the image and activation to begin with (Section 3.3), and we lose even more information by maximizing the noise. So, as a general rule it is better to choose the deepest layer that the edge device can support.

Figure 6 shows the communication and computation costs from the perspective of the edge device as we select deeper cutting points in the networks. As we go deeper, we lose the unnecessary information, therefore ex vivo privacy is monotonically increasing. Computation cost is also monotonically increasing as it's a cumulative function of computation costs of all the preceding layers. Communication cost, on the other hand, is not typically monotonic as the size of the output of a layer can be smaller and also larger than the size of its input. We model the total cost as $Computation \times Communication$ for a specific cutting point. For SVHN, as figure 6a shows, Conv6 is the obvious cutting point choice as it incurs less cost and exhibits more privacy compared to other layers. That is mainly because Conv6 has significantly smaller output than its preceding layers and it substantially brings down the communication cost. For LeNet (figure 6b), we choose Conv2 over Conv0 as we find incurring one percent cost is likely worth the gained privacy level.

4 Related Work

The literature abounds with a variety of attempts to provide greater protection to users' private data in a neural processing system [45, 46, 21, 19]. These efforts span different levels of the system, from training to inference. The majority of these studies [45, 46]; however, have focused on preserving the privacy of contributing users to statistical databases or training models. These techniques tackle the inherent conflict of extracting useful information from a database while protecting private or sensitive data of the individuals from being extracted or leaked [47]. As Table 2 illustrates, the landscape of research in privacy for neural networks can be categorized to the efforts that focus on training or infernce. These categories can be further grouped according to whether or not they require retraining the DNN weights or modifying the model itself (i.e., intrusive). Shredder falls in the category of the techniques that are non-intrusive and target the inference phase. The other technique in this same category, MiniONN [21], uses homomorphic encryption that imposes non-trivial computation overheads making it less suitable for inference on edge. Below, we discuss the most related works, which typically require obtrusive changes to the model, training, or add prohibitively large computation overheads.

Table 2: Privacy Protecting methods in DNNs.

	Non-Intrusive	Intrusive +
Inference	***Shredder*** MiniONN[21]	CryptoNets[19], GAZELLE[48], Arden[49], DPFE[50, 51]
Training/DB	Rappor [15] Apple [14]	Differential Privacy-Based [45, 46], SecureML [52]

+ Intrusive: Model Modification or Retraining

Adding noise for privacy. The idea of noise injection for privacy goes back at least to the very first differential privacy papers [53, 22] where they randomize the result of a query to a database by adding noise drawn from a Laplace distribution. More recently, Wang et al. [49] proposes data nullification and noise injection for private inference.

However, unlike Shredder, they retrain the network. Osia et al. [51, 50] design a private feature extraction architecture that uses principal component analysis (PCA) to reduce the amount of information. Leroux et al. [54] use an autoencoder to obfuscate the data before sending to the cloud, but the obfuscation they use is readily reversible, as they state. We, on the other hand, cast finding the noise as differentiable noise tensor while considering accuracy in the loss function of the optimization that finds the noise.

Trusted execution environments. Several research propose running machine learning algorithms in in trusted execution environments such as Intel SGX [55] and ARM TrustZone [56] to address the same remote inference privacy [57, 58, 59, 60] as well as integrity [57]. However, the privacy model in that research requires users to send their data to an enclave running on a remote servers. In contrast to Shredder, this model still allows the remote server to have access to the raw data and as the new breaches in hardware [1, 2, 61, 62, 63, 64] show, the access can lead to comprised privacy.

Differential privacy. As a mathematical framework, differential privacy [53, 22, 47] was initially proposed to quantify privacy of users in the context of privacy preserving data-mining or statistical databases. To this end, it measures the degree to which the algorithm behaves similarly if an individual record is in or out of the database/training set. This definition gives a robust mathematical guarantee to the question of – given a private training set (or, database entry) as input, how safe is the trained model (or, aggregate database) to publish. Naturally, differential privacy has also been employed in training of deep neural networks [45, 46] where the datasets may be crowdsourced and contain sensitive information. The research on differential privacy is largely in *centralized models*, where users trust a curator who has access to the whole pool of private data [47]. In a more practical model, called local differential privacy, the system does not require users to even trust the curator to inspect their data, even for the purpose of preserving privacy [65, 15, 66, 14]. In this setting, which the system is just collecting data and not performing inference, the data is still scrambled on the edge devices. This scrambled data is then remotely aggregated and just provides an average trend across multiple sources. The existing differential privacy models are in fact solving a fundamentally different problem than Shredder. They are concerned with data collection while Shredder aims to improve privacy during a real-time cloud-enabled inference.

Encryption and cryptographic techniques. Secure multiparty computation (SMC) [16, 17] and homomorphic encryption [21, 19, 48] have also been used as attempts to deal with the privacy on offloaded computation on the cloud [19, 67, 68, 67, 48, 21, 52]. Secure multiparty computation refers to a group of protocols which enable multiple parties to jointly compute a function while each party solely has access to its own part of the input [52, 17]. To establish trust and isolation, SMC relies on compute heavy encryption or obfuscation techniques. To adopt SMC to the privacy problem, recent works [52] assume a two-party secure computation in which the cloud holds a neural network and the client holds an input to the network, typically an image and the communication happens in the encrypted domain. Homomorphic encryption, which can be used to implement SMC, is also used for privacy protection in neural networks. This cryptographic technique allows (all or a subset of) operations to be performed on the encrypted data without the need for decryption. These works [21, 19] suggest the client/edge device encrypt the data (on top of the communication encryption, e.g., ssl) before sending it to the cloud; which it then, performs operations on the encrypted data and returns the output. Nevertheless, this approach suffers from a prohibitive computational and communication cost, exacerbating the complexity and compute-intensivity of neural networks especially on resource-constrained edge devices. Shredder in contrast avoids the significant cost of encryption or homomorphoic data processing.

5 Conclusion

As cloud-based DNNs impact more and more aspects of users' everyday life, it is timely and crucial to consider their impact on privacy. As such, this paper examines the use of noise to reduce the information content of the communicated data to the cloud while still maintaining high levels of accuracy. By casting the noise injection as an optimization for finding a tensor of differentiable elements, we demonstrate Shredder, which strikes an asymmetric balance between accuracy and privacy. Experimentation with multiple DNNs showed that the Shredder can significantly reduce the information content of the communicated data with only 1.46% accuracy loss.

References

[1] 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 *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[2] 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 *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[3] M. A. Cusumano, "Cloud computing and saas as new computing platforms.," *Commun. ACM*, vol. 53, no. 4, pp. 27–29, 2010.

[4] C. Cachin, I. Keidar, A. Shraer, *et al.*, "Trusting the cloud," *Acm Sigact News*, vol. 40, no. 2, pp. 81–86, 2009.

[5] H. R. Motahari-Nezhad, B. Stephenson, and S. Singhal, "Outsourcing business to cloud computing services: Opportunities and challenges," *IEEE Internet Computing*, vol. 10, no. 4, pp. 1–17, 2009.

[6] H. Esmaeilzadeh, E. R. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *IEEE Micro*, vol. 32, pp. 122–134, 2012.

[7] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[8] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[9] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward dark silicon in servers," *IEEE Micro*, vol. 31, pp. 6–15, 2011.

[10] J. Hauswald, M. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. N. Mudge, V. Petrucci, L. Tang, and J. Mars, "Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers," in *ASPLOS*, 2015.

[11] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, (New York, NY, USA), pp. 199–212, ACM, 2009.

[12] NBCNews), "Facebook data harvesting scandal widens to 87 million people," 2019. online accessed May 2019 `https://www.nbcnews.com/tech/tech-news/facebook-data-harvesting-scandal-widens-87-million-people-n862771`.

[13] Axonium), "23andme scandal highlights data privacy concerns shared by axonium and mr koh boon hwee.," 2019. online accessed May 2019 `https://medium.com/@Axonium_org/23andme-scandal-highlights-data-privacy-concerns-shared-by-axonium-and-mr-koh-boon-hwee-dd2e241f1ef2`.

[14] Differential Privacy Team, "Learning with privacy at scale," tech. rep., Apple, 2017. online accessed May 2019 `https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf`.

[15] Úlfar Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 21st ACM Conference on Computer and Communications Security*, (Scottsdale, Arizona), 2014.

[16] A. C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pp. 162–167, Oct 1986.

[17] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A.-R. Sadeghi, G. Scerri, and B. Warinschi, "Secure multiparty computation from sgx," *IACR Cryptology ePrint Archive*, vol. 2016, p. 1057, 2016.

[18] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *In Proc. STOC*, pp. 169–178, 2009.

[19] 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 *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 201–210, JMLR.org, 2016.

[20] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "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, (Berlin, Heidelberg), pp. 45–64, Springer-Verlag, 2013.

[21] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, (New York, NY, USA), pp. 619–631, ACM, 2017.

[22] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, (Berlin, Heidelberg), pp. 265–284, Springer-Verlag, 2006.

[23] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[24] Z. Goldfeld, E. van den Berg, K. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polyanskiy, "Estimating information flow in neural networks," 2018.

[25] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," in *International Conference on Learning Representations*, 2018.

[26] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," 2017.

[27] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[28] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," *2015 IEEE Information Theory Workshop (ITW)*, Apr 2015.

[29] W. Wang, L. Ying, and J. Zhang, "On the relation between identifiability, differential privacy, and mutual-information privacy," *IEEE Transactions on Information Theory*, vol. 62, pp. 5018–5029, Sep. 2016.

[30] J. Liao, L. Sankar, V. Y. F. Tan, and F. du Pin Calmon, "Hypothesis testing under mutual information privacy constraints in the high privacy regime," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 1058–1071, April 2018.

[31] L. Sankar, S. R. Rajagopalan, and H. V. Poor, "Utility-privacy tradeoffs in databases: An information-theoretic approach," *IEEE Transactions on Information Forensics and Security*, vol. 8, pp. 838–852, June 2013.

[32] D. Guo, S. Shamai, and S. Verdú, "Additive non-gaussian noise channels: Mutual information and conditional mean estimation," in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pp. 719–723, IEEE, 2005.

[33] Dongning Guo, S. Shamai, and S. Verdu, "Mutual information and minimum mean-square error in gaussian channels," *IEEE Transactions on Information Theory*, vol. 51, pp. 1261–1282, April 2005.

[34] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2016.

[35] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," *SIGARCH Comput. Archit. News*, vol. 40, pp. 356–367, June 2012.

[36] T. van Laarhoven, "L2 regularization versus batch and weight normalization," *CoRR*, vol. abs/1706.05350, 2017.

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

[38] N. Yann LeCun (Courant Institute and N. Y. Corinna Cortes (Google Labs, "The mnist dataset of handwritten digits." online accessed May 2019 `http://www.pymvpa.org/datadb/mnist.html`.

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

[40] A. Krizhevsky, "Convolutional deep belief networks on cifar-10," 2010.

[41] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. D. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *CoRR*, vol. abs/1312.6082, 2014.

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

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

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

[45] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 909–910, Sep. 2015.

[46] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," 2016.

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

[48] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "Gazelle: A low latency framework for secure neural network inference," in *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, (Berkeley, CA, USA), pp. 1651–1668, USENIX Association, 2018.

[49] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, (New York, NY, USA), pp. 2407–2416, ACM, 2018.

[50] 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," 2017.

[51] S. A. Osia, A. Taheri, A. S. Shamsabadi, M. Katevas, H. Haddadi, and H. R. R. Rabiee, "Deep private-feature extraction," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2018.

[52] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, May 2017.

[53] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "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, (Berlin, Heidelberg), pp. 486–503, Springer-Verlag, 2006.

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

[55] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, (New York, NY, USA), pp. 10:1–10:1, ACM, 2013.

[56] T. Alves and D. Felton, "Trustzone: Integrated hardware and software security," 01 2004.

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

[58] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," 2018.

[59] 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," 2018.

[60] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 619–636, USENIX Association, 2016.

[61] M. Schwarz, M. Lipp, D. Moghimi, J. Van Bulck, J. Stecklina, T. Prescher, and D. Gruss, "ZombieLoad: Cross-privilege-boundary data sampling," *arXiv:1905.05726*, 2019.

[62] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," in *Proceedings of the 27th USENIX Security Symposium*, USENIX Association, August 2018.

[63] O. Weisse, J. Van Bulck, M. Minkin, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, R. Strackx, T. F. Wenisch, and Y. Yarom, "Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution," *Technical report*, 2018.

[64] M. Taram, A. Venkat, and D. Tullsen, "Context-sensitive fencing: Securing speculative execution via microcode customization," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, (New York, NY, USA), pp. 395–410, ACM, 2019.

[65] A. Bittau, U. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld, "Prochlo: Strong privacy for analytics in the crowd," in *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, (New York, NY, USA), pp. 441–459, ACM, 2017.

[66] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (USA), pp. 3574–3583, Curran Associates Inc., 2017.

[67] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," *IACR Cryptology ePrint Archive*, vol. 2017, p. 35, 2017.

[68] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "Xonn: Xnor-based oblivious deep neural network inference." Cryptology ePrint Archive, Report 2019/171, 2019. `https://eprint.iacr.org/2019/171`.