SHREDDER: Learning Noise Distributions to Protect Inference Privacy

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

Sheer amount of computation in deep neural networks has pushed their execution to the cloud. This de facto cloud-hosted inference, however, raises serious privacy concerns as private data is communicated and stored in remote servers. The data could be mishandled by cloud providers, used for unsolicited analytics, or simply compromised through network and system security vulnerability. To that end, this paper devises SHREDDER that reduces the information content of the communicated data without diminishing the cloud's ability to maintain acceptably high accuracy. To that end, SHREDDER learns two sets of noise distributions whose samples, named multiplicative and additive noise tensors, are applied to the communicated data while maintaining the inference accuracy. The key idea is that SHREDDER *learns* these noise distributions offline without altering the topology or the weights of the pre-trained network. SHREDDER repeatedly learns sample noise tensors from the distributions by casting the tensors as a set of trainable parameters while keeping the weights constant. Since the key idea is learning the noise, we are able to devise a loss function that strikes a balance between accuracy and information degradation. To this end, we use self-supervision to train the noise tensors to achieve an intermediate representation of the data that contains less private information. Experimentation with real-world deep neural networks shows that, compared to the original execution, SHREDDER reduces the mutual information between the input and the communicated data by 66.90%, and yields a misclassification rate of 94.5% over private labels, significantly reducing adversary's ability to infer private data, while sacrificing only 1.74% loss in accuracy without any knowledge about the private labels. Such a high degree of misclassification for private labels, another practical measure to asses privacy, shows that SHREDDER is an effective initial step.

Introduction

Online services that utilize the cloud infrastructure are now ubiquitous and dominate the IT industry (Cusumano 2010). The limited computation capability of edge devices and the increasing processing demand of learning models (Jordan and Mitchell 2015) has naturally pushed most of the computation to the cloud (Hardavellas et al. 2011). These cloud 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 warehouse-scale cloud computing model brings unprecedented capabilities, it significantly compromises user privacy. On the cloud, data can be compromised through side-channel hardware attacks (e.g., Spectre (Kocher et al. 2019)) or deficiency in the software stack (Ristenpart et al. 2009). But even in the absence of such attacks, the service provider can share the data with business partners (NBC News 2018) or government agencies (Axonium 2018). Although the industry has adopted privacy techniques for data collection and model training (Differential Privacy Team 2017; lfar Erlingsson, Pihur, and Korolova 2014), scant attention has been given to the privacy of users who increasingly rely on online services for inference.

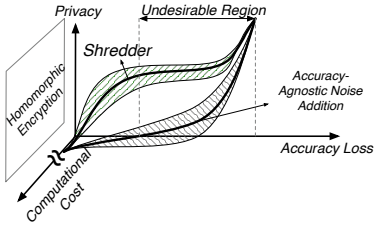


Figure 1: The landscape of research in inference privacy and how SHREDDER fits in the picture.

As Figure 1 illustrates, researchers have attempted to grapple with this problem by employing cryptographic techniques such as multiparty execution (Yao 1986; Bahmani et al. 2016) and homomorphic encryption (Liu et al. 2017) 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. Moreover, such approaches also add the burden of encryption and decryption to the already constrained edge devices despite the computational limit being the main incentive of offloading the inference to the cloud.

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. In this model a portion of the DNN inference is executed on the edge and the rest is delegated to the cloud and noise is applied to the intermediate activation tensor that is sent to the remote servers from the first portion. 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 uses a gradient-based learning process to extract the distributions of noise from different learned samples. To learn the noise samples, we define a loss function that incorporates both accuracy and the distance of the intermediate representation of the same public label. We use a self-supervised learning process to learn representations (i.e., noise-injected activations) to have larger distance for different labels and closer distance for the same labels. This feature enables SHREDDER to focus the noise-injected representations on the public labels and diminish the classification capability of the DNN on possible unseen private labels. *In this whole process, the only trainable parameters are the noise tensors while the weights and topology of the network is kept intact and the possible private labels are not exposed to* SHREDDER. 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 collection of significantly smaller noise tensors through a gradient-driven optimization process that are converted to distributions.

The main insight is that the noise can be seen as an added trainable set of parameter probabilities that can be discovered through repetition of an end-to-end self-supervised training process. We have devised the noise training loss such that it exposes an asymmetric tradeoff between accuracy and privacy as depicted in Figure 1. 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 (Dwork and Roth 2014) 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) 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 66.90% compared to the original execution with only 1.74% accuracy loss. We also evaluate the performance of a model trained with this self-supervised process, with no private labels available, against private label classification, and show its effectiveness in causing average misclassification rate of 94.5% with 1.5% degradation in public (primary) label classification. Compared to another privacy protection method which is aware of the private labels, DPFE, SHREDDER provides $1.6\times$ more loss in mutual information, while giving only 2% less misclasification rate, on VGG-16 neural network for face identification task. 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. We also have uploaded the SHREDDER code on the submission website.

Shredder: Noise Distribution Learning Framework

This section delves deeper into the details of SHREDDER, starting from giving an overview of the framework and its two disjoint phases: 1) an offline noise learning phase and 2) an inference phase.

Phase I: learning the noise distributions. The first phase takes in the DNN architecture, the pretrained weights, and a training dataset. The training dataset is the same as the one used to train the DNN. The output of this phase is a collection of 100 tuples of multiplicative noise distributions and additive noise distributions each coupled with the order for the elements of that noise tensor (to maintain relative order of elements and preserve accuracy) for the inference phase. This phase also determines which layer is the optimal choice for cutting the DNN to strike the best balance between computation and communication while considering privacy. The deeper the cutting point, the higher the privacy level, given a fixed level of loss in accuracy. This is due to the abstract representation of data in deeper layers of neural networks (Yosinski et al. 2014), which cause for less communicated information to begin with, giving the framework a head start. So, as a general rule it is better to choose the deepest layer that the edge device can support. To find the best cutting point in term of communication and computation costs, we conducted experiments commensurate with those of Neurosergeon (Kang et al. 2017) and examine each layer in terms of total time of computation and communication and pick the lowest. In addition, this phase outputs the mean accuracy and a margin of error for its collection of distributions. This mean and margin are achieved by experimentation on a held-out portion of the training set.

Phase II: shredder inference and noise sampling. In this phase, for each inference pass (each time a data is given to the neural network) the collection of tuples, from phase I, is sampled for a tuple of multiplicative noise distribution, additive noise distribution and element orders. Then, the noise distributions (which are both Laplace distributions) are sampled to populate the additive and multiplicative noise tensors, which have the same dimensions as the intermediate activation. Then, the elements of both noise tensors are rearranged, so as to match the saved order for that distribution. For this, the sampled elements are all sorted, and are then replaced according to the saved order of indices from the learning phase. This process makes predicting the noise tensor non-trivial for the adversary, since the noise for each input data is generated **stochastically**. The multiplicative noise, which is merely a scale and has the same shape as the intermediate activations, is applied to intermediate acti-

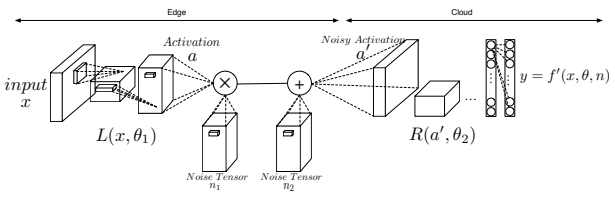


Figure 2: Noise injection in SHREDDER.

vations followed by the generated additive noise and the final noisy activation is sent from the edge device to the cloud where the rest of the inference will take place.

Trainable Noise Tensors

Given a pre-trained network $f(x, \theta)$ with K layers and pre-trained 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 f' in this image is the noisy network output (noisy logits) which are the outputs of the last layer of the neural network before going through softmax layer. The user provides input x to the local network, and an intermediate activation tensor $a = L(x, \theta_1)$ is produced. Then, a multiplicative noise tensor n_1 and an additive noise tensor n_2 are applied to the output of the first part, $a' = n_1 \times a + n_2$. This a' is then communicated to the cloud where $R(a', \theta_2)$ is computed on noisy data and produces the result $y = f'(x, \theta, n_1, n_2)$ that is sent back to the user.

The objective is to find the noise tensors n_1, n_2 that minimize our loss function (discussed in later sections), while the rest of the model parameters are fixed. To be able to do this through a gradient based method of optimization, we must find the $\partial y/\partial n_1$ and $\partial y/\partial n_2$. For the former, we have:

$$\frac{\partial y}{\partial n_1} = \frac{\partial f'(x, \theta, n_1, n_2)}{\partial n_1} = \frac{\partial R((n_1 \times a + n_2), \theta_2)}{\partial n_1}$$
$$= \frac{\partial layer_{K-1}}{\partial Layer_{K-2}} \times ... \times \frac{\partial Layer_{c+1}}{\partial (n_1 \times a + n_2)} \times \underbrace{\frac{\partial (n_1 \times a + n_2)}{\partial n_1}}_{= \frac{\partial n_1 \times a}{\partial n_1} = a} \quad (1)$$

Since $L(x, \theta_1)$ is not a function of n_1 or n_2, it is not involved in the backpropagation. The same math can be applied to get $\partial y/\partial n_2$. Gradient of R is also computed through chain rule as shown above. Therefore, the output is differentiable with respect to the noise tensor and gradient based methods can be employed to solve the optimization problem.

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

$$I(x; a) = \int_x \int_a p_{x,a} log_2 \frac{p_{x,a}}{p_x p_a} dx da. \quad (2)$$

Mutual information has been widely used in the literature for both understanding the behavior of neural networks (Saxe et al. 2018), and also to quantify information leakage in anonymity systems in the context of databases (Wang, Ying, and Zhang 2016; Liao et al. 2018; Sankar, Rajagopalan, and Poor 2013). We also use the reverse of mutual information ($1/MI$) as a notion of privacy and call it ex vivo privacy. In our setting, we quantify the information between the user-provided input and the intermediate activations that are sent to the cloud. Mutual information is considered an information theoretic notion. Hence, it quantifies the average amount of information about the input (x) that is contained in the intermediate activations (a).

In Vivo Notion of Privacy

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, we introduce an in vivo notion of privacy which is the reverse of signal to noise ratio ($1/SNR$), as a proxy for our ex vivo notion of privacy, to monitor privacy level during training. Mutual information is shown to be a function of SNR in noisy channels (Guo, Shamai, and Verdú 2005).

Loss Function and Self-Supervised Learning

The objective of the optimization is to find the noise distributions in such a way as to minimize $I(x, a')$ and at the same time maintains the accuracy. 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 tolerance of the network to perturbations yields such behavior. Thus, we use self-supervision to train the noise tensors to achieve an intermediate representation of the data that contains less private information. In our problem definition, the framework is not aware of what data is considered private. It only knows the primary classification task of the network. Therefore, the framework assumes that anything except the primary labels is excessive information. In other words, the training process uses the information it has, supervises itself, and learns a representation that only contains the necessary information without access to the private labels. To make this possible, SHREDDER attempts at decreasing the distance between representations (intermediate activations) of inputs with the same primary labels and increasing the distance between those with different labels. This approach trains the noise tensors as if the framework is speculating what information may be private, and it tries to remove it, but using only the public primary labels.

In the evaluation section we asses the performance this self-supervised process, against private label classification, and show its effectiveness in causing high misclassification rates for them. As mentioned before, SHREDDER uses two noise tensors, both of which are the same size as the activations they are being multiplied by (scaled) and added to. The number of elements in these noise tensors equals to the number of trainable parameters in our method. The rest of the model parameters are all fixed.

During conventional training at each iteration, a batch of training data is selected, fed through the network. Then,

using a given loss function, back-propagation and gradient based methods, the trainable parameters are updated. SHREDDER's algorithm, however, as shown in Algorithm 1, modifies this by choosing a second random batch of training data, passing it through the first partition of the neural network, $L(x, \theta_1)$ in Figure 2, and then applying both multiplicative and additive noise tensors. This yields a'_i and a'_j for members of the main batch and random batch, respectively. Then, the $L2$ distance between the representations for corresponding members of the main batch and the random batch is computed. If the corresponding elements in each of the batches have the same primary labels, the distance between their representations should have a positive coefficient, since we would want to decrease it. If the primary labels are different, the distances would get a negative coefficient before being used in the loss function. Hence, a loss function with the formulation below is given:

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) + \lambda(\sum_{(i,j):Y_i=Y_j} |a'_i - a'_j|^2 + \sum_{(i,j):Y_i \neq Y_j} (c - |a'_i - a'_j|^2)) \tag{3}$$

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 minimizes the distance between intermediate activations with same labels while maximizing the distance between those with different labels. i, j are iterators over the main batch and random batch members, respectively and Y is the primary label for that batch member. λ and c are hyperparameters that should be tuned for each network. 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 usually use 0.01, 0.001 and 0.0001.

SHREDDER initializes the multiplicative noise tensor to 0 and the additive 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 initial noise (privacy) of our model.

Algorithm 1 Shredder's training process.

```
procedure TRAINING(model, training_set,λ, c)
    for all (data, labels) ∈ training_set do
        (data_rand, labels_rand) ← random(training_set)
        output ← model.local(data)
        output ← model.noise(output)
        output_rand ← model.local(data_rand)
        output_rand ← model.noise(output_rand)
        distance ← |output_rand-output|^2
        positive ← (labels == labels_rand)
        negative ← (labels != labels_rand)
        positive ← sum(distance × positive)
        negative ← sum(distance × negative)
        logits ← model(data)
        loss ← CrossEntropy(logits, labels) + λ(pos+(c-neg))
        update_noise_tensors(loss)
    end for
end procedure
```

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 intractable. Thus, SHREDDER uses an in vivo notion of privacy which uses (SNR) as a proxy to MI (Guo, Shamai, and Verdú 2005). In other words, SHREDDER uses SNR to monitor privacy during training. 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, which is $a' - a$, using the notation from Figure 2. We use this to estimate the relative privacy achieved for each set of hyperparameters, during parameter tuning.

Extracting Distributions and Element Orders

During training, the model is constantly tested on a held out portion of the training set. When the accuracy goes higher than a given threshold (the amount the user is willing to compromise) the training is halted and SHREDDER proceeds to distribution extraction. We use maximum likelihood estimation of the distribution parameters, i.e, loc and scale, to fit each learned noise tensor to a Laplace distribution. It's worth mentioning that this stage is executed offline. The parameters for these Laplace distributions are saved. The element orders of the noise tensor are also saved. By the element orders, the sorted indices of the elements of the flattened noise tensor is meant. For instance, if a tensor looks like $[[3.2, 4.8], [7.3, 1.5]]$, it's flattened version would be $[3.2, 4.8, 7.3, 1.5]$, and the sorted which is what the collector saves would be $[2, 1, 0, 3]$.

Empirical Evaluation

We evaluate the accuracy-privacy trade-off, the noise training process with our loss function, and a comparison to another privacy protection mechanism.

Experimental methodology. Mutual Information (MI) is calculated using the Information Theoretical Estimators Toolbox's (Szabó 2014) Shannon Mutual Information with KL Divergence. Due to the high dimensionality of the tensors (especially large images) mutual information estimations are not very accurate. That is why SHREDDER uses feature selection methods for large networks to decrease dimensionality for MI measurements. The first step is using only one channel, from the three input channels, since the the other two channels hold similar information. The second step is removing features with single unique value, and also features with collinear coefficient of higher than 0.98. This helps reduce the dimensionality gravely. In the results reported in upcoming sub-sections, MI is calculated over the shuffled test sets on MNIST dataset for LeNet, CIFAR-10 dataset for CIFAR-10, SVHN dataset for SVHN, ImageNet dataset for AlexNet, a subset of 24 celebrity faces from VGG-Face (Parkhi, Vedaldi, and Zisserman 2015) for VGG-16 and 20Newsgroups (Lang 2008) for a 5 layer DNN.

These photos were shuffled through and chosen at random. The primary classification task for VGG-16 is modified to be gender classification of the celebrity faces. 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.

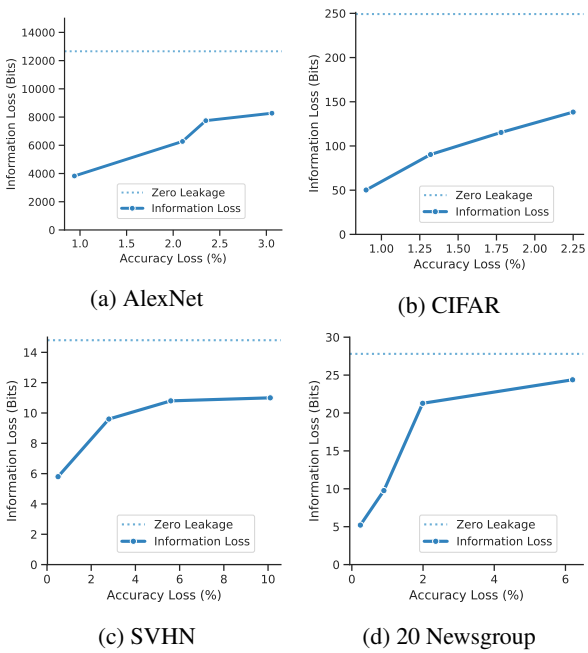
Table 1 summarizes our experimental results. It is shown that on the networks, SHREDDER can achieve on average 66.90% loss in information while inducing 1.74% loss in accuracy with an average margin of error of $\pm 0.22\%$. The table also shows that it takes SHREDDER a short time to train the noise tensor, for instance on AlexNet it is 0.2 epoch. SHREDDER has $\pm 0.22\%$ trainable parameters compared to another method, DPFE (Osia et al. 2018).

Accuracy-privacy trade-off. There is a trade-off between the noise that is applied to the network, and its accuracy. As shown in Figure 1, SHREDDER attempts to increase privacy (noise) while keeping the accuracy intact. Figures 3 and 4 show the level of privacy that can be obtained by losing a given amount of accuracy for LeNet, CIFAR-10, SVHN, AlexNet, VGG-16 and 20Newsgroups. The primary classification task for AlexNet, CIFAR, SVHN and 20Newsgroups is their conventional use-case, and we do not define a private task. However, in Figure 4 for LeNet and VGG-16, we have modified the primary classification task to learn whether the number in the image is greater than five, and to detect the gender of faces in the images, respectively. The private tasks are defined as classifying what the numbers in the images are exactly, and identifying the faces in the celebrity images.

Figures 3 and 4 show the number of mutual information bits that are lost from the original activation using our method, on the left side Y axis. For LeNet and VGG-16, in Figures 4a and 4b, there is a second Y axis which shows the normalized (over pre-trained accuracy) misclasification rate for the private task. The cutting point of the networks is their last convolution layer. The dotted *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 blue connected 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 Figures 3 and 4. Once the excess information is removed, the residue is 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 significant accuracy loss. The extreme to this case can be seen in Figures4a and 3d, where approaching the *Zero Leakage* line causes over 7% loss. The misclassification rate for private tasks in LeNet and VGG-16 is increasing, as more information is lost.

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 eliminate excess information. Figure 5 shows part of the training process on VGG-16 for gender classification on the VGG-Face dataset, cut from its last convolution layer. The multiplicative noise tensor was initialized to 1 and the additive to 0, so as to better show the effectiveness of the method, on a pre-trained network with no initial noise to help privacy. The black line shows the accuracy of the private task, identifying which face belongs to whom.

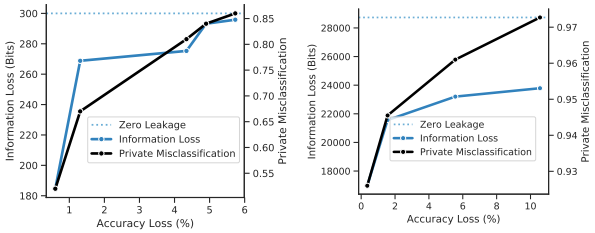
As Figure 5 shows in black, the in vivo notion of privacy ($1/SNR$) is increasing as the training moves forward. The accuracy of the private task, however, is decreasing while the accuracy of the primary task is relatively stable. It is note-



(a) AlexNet (b) CIFAR

(c) SVHN (d) 20 Newsgroup

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) LeNet (b) VGG16

Figure 4: Accuracy-Privacy trade-off for VGG-16 and LeNet, cut from their last convolution layer. The zero leakage line shows the original mutual information between input images and activations at the cutting point. The right-hand Y axis shows the normalized (over pre-trained accuracy) misclasification rate of private labels.

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

Benchmark	LeNet	CIFAR	SVHN	Alexnet	VGG-16	20Newsgroups	Average
Original Mutual Information	300.04	249.24	14.85	12661.52	28732.21	27.8	–
Shredded Mutual Information	31.2	115.3	5.1	5312.53	7168.75	6.52	–
Mutual Information Loss	**89.60%**	**46.27%**	**64.86%**	**58.05%**	**75.05%**	**76.55%**	**66.90%**
Accuracy Loss	**1.31%**	**1.78%**	**1.72%**	**1.95%**	**2.18%**	**1.99%**	**1.74%**
Margin of Error	±0.59%	±0.57%	±0.12%	±0.01%	±0.04%	±0.02%	±0.22%
SHREDDER's Learnable Params / DPFE(Osia et al. 2018)	0.39%	1.29%	0.09%	0.03%	0.04%	0.10%	0.32%
Number of Epochs of Training	5.2	1.8	2.2	0.2	5.7	5.2	2.2

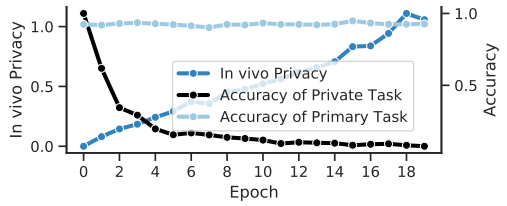
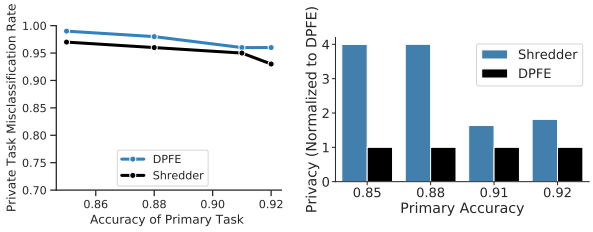


Figure 5: In vivo notion of privacy, normalized accuracy of private task and primary task per epoch of training on VGG-16, when the last convolution layer is the cutting point.



(a) Misclasification rate of private labels

(b) Normalized privacy improvement over DPFE

Figure 6: Normalized misclasification rate of private labels (identity) and privacy improvement comparison with DFFE for different accuracy levels of the primary task (gender classification) over VGG-16 on VGG-Face dataset.

worthy that the accuracy numbers are normalized to the pre-trained accuracies for private and primary tasks.

Comparison with DPFE. Deep Private Feature Extraction (DPFE) (Osia et al. 2018) is a privacy protection mechanism that aims at obfuscating given private labels. For these experiments, VGG-16 network is used with celebrity images which is the same setup used in the DPFE paper. DPFE partitions the network in two partitions, first partition to be deployed on the edge and the second for the cloud. It also modifies the network architecture by adding an auto-encoder in the middle, to reduce dimensions, and then re-training the entire network with its loss function. DPFE's loss function assumes knowledge of private labels, and tries to maintain accuracy while removing the private labels through decreasing the distance between intermediate activations with different **private labels** and increasing the distance between intermediate activations of inputs with the same **private label**. After training, for each inference, a randomly generated noise is added to the intermediate results on the fly. DPFE can only be effective if the user knows what s/he wants to protect against, whereas SHREDDER offers a more general approach that tries to eliminate any information that is irrelevant to the primary task. Table 1 has a row which compares the number of trainable parameters for SHREDDER with DPFE and it can be seen that SHREDDER's parameters are extremely lower than DPFE's. DPFE also incurs extra computation overhead with its embedded auto-encoder and is intrusive to the model, since it modifies the architecture and parameters, and needs to re-deploy the model. Whereas, SHREDDER does not modify the already deployed models, it just multiplies and adds the noise.

To run experiments, the intermediate outputs of the networks are fed to two classifiers, for gender and identity. Figure 6a compares DPFE to SHREDDER in terms of the missclasification rate of the private task, i.e. the identity classification (identity compromise) for given levels of accuracy for the main task, which is the same metric used in (Osia et al. 2018). SHREDDER causes negligibly less misclassification rate, which can be attributed to its unawareness of the private labels, and also its fewer number of trainable parameters which limit its control on the training.

Figure 6b shows privacy improvement of SHREDDER over DPFE. It can be inferred from the Figure that SHREDDER performs better, since it has a more general approach which scrambles more overall information. DPFE takes an approach which is directed at a specific goal, which impedes it from providing privacy for aspects other than the private task. To compare the methods more intuitively, if there are images of people with their surrounding, DPFE would make sure that the identities of these people are hidden, but would not eliminate the background, or any other information. SHREDDER however, removes information in all aspects, possibly eliminating the background as well.

Related Work

Several efforts attempt to provide greater protection for private data in a neural systems (Xiao et al. 2017; Shokri and Shmatikov 2015; Abadi et al. 2016; Liu et al. 2017; Dowlin et al. 2016). As Table 2 illustrates, the research in privacy for neural networks can be categorized to the efforts that focus on training or inference. The majority of these studies (Shokri and Shmatikov 2015; Abadi et al. 2016); however, focus on preserving the privacy of contributing users to training models or statistical databases. (Bonawitz et al. 2017), for instance, introduces a privacy-preserving protocol for federated learning. These categories in Table 2 can be further grouped according to whether or not they re-

Table 2: Privacy Protecting methods in DNNs.

		Privacy protecting methods
Inference	Non-Intrusive	SHREDDER
		MiniONN(Liu et al. 2017)
	Intrusive	Arden (Wang et al. 2018)
		DPFE (Osia et al. 2017; Osia et al. 2018)
		CryptoNets (Dowlin et al. 2016)
		GAZELLE (Juvekar, Vaikuntanathan, and Chandrakasan 2018)
Training	Non-Intrusive	Rappor (Ifar Erlingsson, Pihur, and Korolova 2014)
		Apple (Differential Privacy Team 2017)
	Intrusive	SecureML (Mohassel and Zhang 2017)
		With Differential Privacy (Abadi et al. 2016)
		With Differential Privacy (Shokri and Shmatikov 2015)

quire retraining the DNN weights or modifying the model itself (i.e., intrusive). SHREDDER falls in the category of non-intrusive techniques that target inference. The other technique in this same category, MiniONN (Liu et al. 2017), 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.

Adding noise for privacy. The idea of noise injection for privacy goes back at least to the very first differential privacy papers (Dwork et al. 2006) where they randomize the result of a query to a database by adding noise drawn from a Laplace distribution. More recently, Wang et al. (Wang et al. 2018) proposes data nullification and noise injection for private inference. However, unlike SHREDDER, they retrain the network. Osia et al. (Osia et al. 2018; Osia et al. 2017) design a private feature extraction architecture that uses principal component analysis (PCA) to reduce the amount of information. Leroux et al. (Leroux et al. 2018) 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.

Self-supervised learning. Self-supervised learning is becoming more and more prevalent (Jamaludin, Kadir, and Zisserman 2017). (Hendrycks et al. 2019) finds that self-supervision can help robustness, including robustness to adversarial examples, label corruption, and common input corruptions. (Owens and Efros 2018) uses self-supervision to learn representations for applications of sound source localization, audio-visual action recognition and on/off-screen audio source separation. SHREDDER takes self-supervision to a new direction, privacy protection on edge devices.

Trusted execution environments. Several research propose running machine learning algorithms in in trusted execution environments such as Intel SGX and ARM TrustZone to address the same remote inference privacy (Tramer and Boneh 2019; Hunt et al. 2018) as well as integrity (Tramer and Boneh 2019). 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 (Kocher et al. 2019) show, the access can lead to comprised privacy.

Differential privacy. As a mathematical framework, differential privacy (Dwork and Roth 2014) 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 (Dwork and Roth 2014). Naturally, differential privacy has also been employed in training of deep neural networks (Shokri and Shmatikov 2015; Abadi et al. 2016) where the datasets may be crowdsourced and contain sensitive information. 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) (Bahmani et al. 2016) and homomorphic encryption (Liu et al. 2017; Dowlin et al. 2016; Juvekar, Vaikuntanathan, and Chandrakasan 2018) have also been used as attempts to deal with the privacy on offloaded computation on the cloud (Dowlin et al. 2016; Juvekar, Vaikuntanathan, and Chandrakasan 2018; Liu et al. 2017; Mohassel and Zhang 2017). 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 (Liu et al. 2017; Dowlin et al. 2016) 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.

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. This paper takes a self-supervised learning approach in using 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 as a gradient-based learning process for finding the distributions of noise tensors, SHREDDER strikes an asymmetric balance between accuracy and privacy, that is evaluated using a set of real-world DNNs.

References

[Abadi et al. 2016] Abadi, M.; Chu, A.; Goodfellow, I. J.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep

learning with differential privacy. In *In Proceedings ACM Conference on Computer and Communications Security (CCS)*.

[Axonium 2018] Axonium. 2018. 23andme scandal highlights data privacy concerns shared by axonium and mr koh boon hwee.

[Bahmani et al. 2016] Bahmani, R.; Barbosa, M.; Brasser, F.; Portela, B.; Sadeghi, A.-R.; Scerri, G.; and Warinschi, B. 2016. Secure multiparty computation from sgx. *IACR Cryptology ePrint Archive*.

[Bonawitz et al. 2017] Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H. B.; Patel, S.; Ramage, D.; Segal, A.; and Seth, K. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of CCS*.

[Cusumano 2010] Cusumano, M. A. 2010. Cloud computing and saas as new computing platforms. *Commun. ACM* 53(4):27–29.

[Dowlin et al. 2016] Dowlin, N.; Gilad-Bachrach, R.; Laine, K.; Lauter, K.; Naehrig, M.; and Wernsing, J. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*.

[Dwork and Roth 2014] Dwork, C., and Roth, A. 2014. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* 9.

[Dwork et al. 2006] Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography (TCC)*.

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

[Hardavellas et al. 2011] Hardavellas, N.; Ferdman, M.; Falsafi, B.; and Ailamaki, A. 2011. Toward dark silicon in servers. *IEEE Micro* 31.

[Hendrycks et al. 2019] Hendrycks, D.; Mazeika, M.; Kadavath, S.; and Song, D. 2019. Using self-supervised learning can improve model robustness and uncertainty. *CoRR* abs/1906.12340.

[Hunt et al. 2018] Hunt, T.; Song, C.; Shokri, R.; Shmatikov, V.; and Witchel, E. 2018. Chiron: Privacy-preserving machine learning as a service. *CoRR* abs/1803.05961.

[Jamaludin, Kadir, and Zisserman 2017] Jamaludin, A.; Kadir, T.; and Zisserman, A. 2017. Self-supervised learning for spinal mris. *CoRR* abs/1708.00367.

[Jordan and Mitchell 2015] Jordan, M. I., and Mitchell, T. M. 2015. Machine learning: Trends, perspectives, and prospects. *Science* 349(6245).

[Juvekar, Vaikuntanathan, and Chandrakasan 2018] Juvekar, C.; Vaikuntanathan, V.; and Chandrakasan, A. 2018. Gazelle: A low latency framework for secure neural network inference. In *Proceedings of USENIX Conference on Security Symposium (SEC)*.

[Kang et al. 2017] Kang, Y.; Hauswald, J.; Gao, C.; Rovinski, A.; Mudge, T.; Mars, J.; and Tang, L. 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *In Proceedings of ACM Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

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

[Lang 2008] Lang, K. 2008. The 20 newsgroups data set. online accessed July 2019 http://qwone.com/~jason/20Newsgroups/.

[Leroux et al. 2018] Leroux, S.; Verbelen, T.; Simoens, P.; and Dhoedt, B. 2018. Privacy aware offloading of deep neural networks. *CoRR* abs/1805.12024.

[Liao et al. 2018] Liao, J.; Sankar, L.; Tan, V. Y. F.; and du Pin Calmon, F. 2018. Hypothesis testing under mutual information privacy constraints in the high privacy regime. *IEEE Transactions on Information Forensics and Security* 13(4).

[Liu et al. 2017] Liu, J.; Juuti, M.; Lu, Y.; and Asokan, N. 2017. Oblivious neural network predictions via minionn transformations. In *CCS*.

[Mohassel and Zhang 2017] Mohassel, P., and Zhang, Y. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *Proceedings of S&P*.

[NBC News 2018] NBC News. 2018. Facebook data harvesting scandal widens to 87 million people. online accessed May 2019 https://www.nbcnews.com/tech/tech-news/facebook-data-harvesting-scandal-widens-87-million-people-n862771.

[Osia et al. 2017] Osia, S. A.; Shamsabadi, A. S.; Sajadmanesh, S.; Taheri, A.; Katevas, K.; Rabiee, H. R.; Lane, N. D.; and Haddadi, H. 2017. A hybrid deep learning architecture for privacy-preserving mobile analytics. *CoRR* abs/1703.02952.

[Osia et al. 2018] Osia, S. A.; Taheri, A.; Shamsabadi, A. S.; Katevas, M.; Haddadi, H.; and Rabiee, H. R. R. 2018. Deep private-feature extraction. *IEEE Transactions on Knowledge and Data Engineering*.

[Owens and Efros 2018] Owens, A., and Efros, A. A. 2018. Audio-visual scene analysis with self-supervised multisensory features. *CoRR* abs/1804.03641.

[Parkhi, Vedaldi, and Zisserman 2015] Parkhi, O. M.; Vedaldi, A.; and Zisserman, A. 2015. Deep face recognition. In *British Machine Vision Conference*.

[Differential Privacy Team 2017] Differential Privacy Team. 2017. Learning with privacy at scale. Technical report, Apple.

[Ristenpart et al. 2009] Ristenpart, T.; Tromer, E.; Shacham, H.; and Savage, S. 2009. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proceedings of CCS*.

[Sankar, Rajagopalan, and Poor 2013] Sankar, L.; Rajagopalan, S. R.; and Poor, H. V. 2013. Utility-privacy tradeoffs in databases: An information-theoretic approach. *IEEE Transactions on Information Forensics and Sec.* 8(6).

[Saxe et al. 2018] Saxe, A. M.; Bansal, Y.; Dapello, J.; Advani, M.; Kolchinsky, A.; Tracey, B. D.; and Cox, D. D. 2018. On the information bottleneck theory of deep learning. In *In Proceedings of ICML*.

[Shokri and Shmatikov 2015] Shokri, R., and Shmatikov, V. 2015. Privacy-preserving deep learning. In *In Proceedings of CCS*.

[Szabó 2014] Szabó, Z. 2014. Information theoretical estimators toolbox. *Journal of Machine Learning Research* 15.

[Tramer and Boneh 2019] Tramer, F., and Boneh, D. 2019. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *In Proceedings of ICLR*.

[Wang et al. 2018] Wang, J.; Zhang, J.; Bao, W.; Zhu, X.; Cao, B.; and Yu, P. S. 2018. Not just privacy: Improving performance of private deep learning in mobile cloud. In *Proceedings of Conference on Knowledge Discovery & Data Mining (KDD)*.

[Wang, Ying, and Zhang 2016] Wang, W.; Ying, L.; and Zhang, J. 2016. On the relation between identifiability, differential privacy,

and mutual-information privacy. *IEEE Transactions on Information Theory* 62(9).

[Xiao et al. 2017] Xiao, Q.; Li, K.; Zhang, D.; and Xu, W. 2017. Security risks in deep learning implementations. *CoRR* abs/1711.11008.

[Yao 1986] Yao, A. C. 1986. How to generate and exchange secrets. In *Symposium on Foundations of Computer Science (SFCS)*.

[Yosinski et al. 2014] Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *Proceedings of NIPS*.

[lfar Erlingsson, Pihur, and Korolova 2014] lfar Erlingsson; Pihur, V.; and Korolova, A. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of CCS*.