Practical Privacy-Preserving MLaaS: When Compressive Sensing Meets Generative Networks

Jia Wang[1], Wuqiang Su[1], Zushu Huang[1], Jie Chen [1], Chengwen Luo[2], Jianqiang Li[2] *

[1] College of Computer Science and Software Engineering, Shenzhen University
[2] National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University
3688 Nanhai Avenue, Shenzhen, Guangdong Province, China
{jia.wang, suwuqiang2019}@szu.edu.cn, 2060271076@email.szu.edu.cn, {chenjie, chengwen, lijq}@szu.edu.cn

Abstract

The Machine-Learning-as-a-Service (MLaaS) framework allows one to grab low-hanging fruit of machine learning techniques and data science, without either much expertise for this sophisticated sphere or provision of specific infrastructures. However, the requirement of revealing all training data to the service provider raises new concerns in terms of privacy leakage, storage consumption, efficiency, bandwidth, etc. In this paper, we propose a lightweight privacy-preserving MLaaS framework by combining Compressive Sensing (CS) and Generative Networks. It's constructed on the favorable facts observed in recent works that general inference tasks could be fulfilled with generative networks and classifier trained on compressed measurements, since the generator could model the data distribution and capture discriminative information which are useful for classification. To improve the performance of the MLaaS framework, the supervised generative models of the server are trained and optimized with prior knowledge provided by the client. In order to prevent the service provider from recovering the original data as well as identifying the queried results, a noise-addition mechanism is designed and adopted into the compressed data domain. Empirical results confirmed its performance superiority in accuracy and resource consumption against state-of-the-art privacy-preserving MLaaS frameworks.

Introduction

The past decades have witnessed the proliferation of AI-enhanced systems in various application domains. The promising performance achieved by recent machine learning approaches have motivated users in different businesses to apply machine learning algorithms to their own datasets. The current machine learning frameworks, however, are usually complicated to non-expert users due to large number of parameters required for configuration and lack of general knowledge on how machine learning works. The increasing demand on machine learning for non-expert users has nourished a new computing paradigm known as the *Machine Learning as a Service (MLaaS)* (Ribeiro, Grolinger, and Capretz 2015), in which data uploading interfaces are provided while learning details are usually appeared as black boxes for users. Since MLaaS provides a convenient way for users to upload training data and train models for real-time classification tasks, it has started to gain popularities and offered services by Google (Google 2020), Microsoft (Microsoft 2020), Amazon (Amazon 2020), and etc.

In typical MLaaS framework, users are allowed to label and upload their own data for their customized model training. After the models are trained, users can upload queries and the cloud returns classification results in realtime. In this way, the authors do not need to understand the details of the model training process and can leverage the cloud computation powers in both offline training and online classification phases. This significantly lowers the barriers for non-expert users in using the machine learning services, however, it also greatly increases the potential risks of user privacy leakage. As shown in Figure 1, in the offline training phase, the plain training data uploaded to the cloud has no privacy protection guarantees and faces arbitrary attacks with the existence of malicious cloud admins. In the online classification phase, both the queries containing users' realtime data and classification results (i.e., labels) are private to users, and without efficient privacy protection approaches users will be put under a circumstances with severe privacy leakage risks. Therefore, it is crucial for MLaaS providers to design a mechanism for efficient privacy protection along with the basic machine learning services.

To achieve practical privacy-preserving MLaaS, several requirements exist. First, the computational burden imposed by the privacy protection mechanisms should be minimized. Second, the communication costs should also be minimized to meet the increasing demand of MLaaS services. Last but not the least, the classification performance should meet the application requirements and should not sacrifice much on the classification accuracy. Current solutions on privacy preserving machine learning fail to meet all the above requirements. For example, Secure Multi-party Computation (SMC) (Knott et al. 2021)and Federated Learning (FL) (Li et al. 2021)are all promising methods in relative fields. However, SMC and FL involve model training and multi-round interactions among participants, which doesn't align with the MLaaS premise where clients often lack expertise in model design and training capabilities. Homomorphic encryption (Acar et al. 2018) for machine learning (Takabi, Hesamifard, and Ghasemi 2016) provides learning capabilities over encrypted data, and has become one promis-

*Corresponding author.

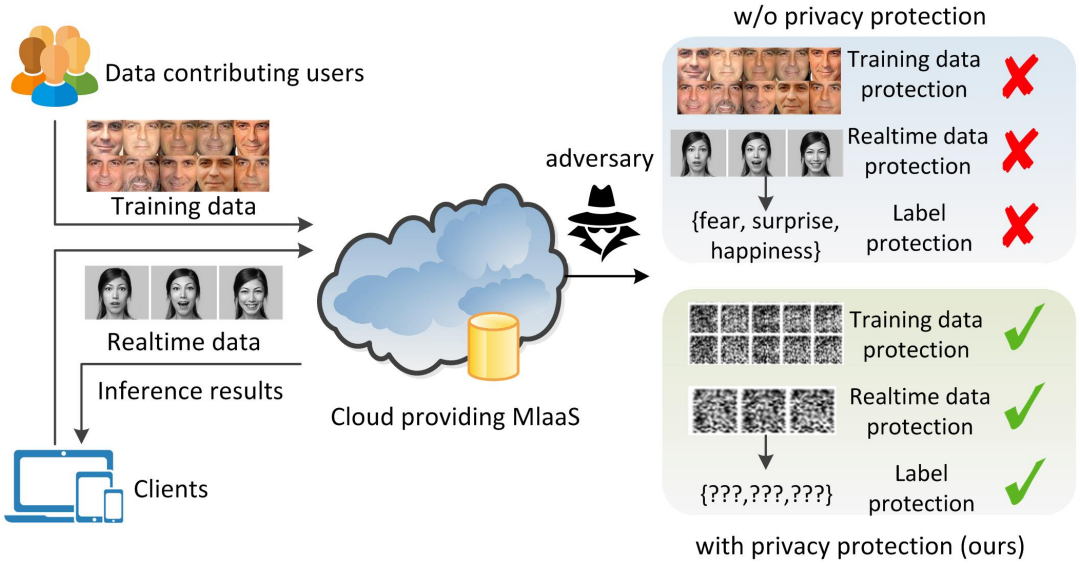


Figure 1: MLaas poses significant risks on training data protection, realtime data protection, and label protection. Our methods provides protections on all three parts while ensuring the usability of machine learning services.

ing privacy-preserving machine learning approach. However, homomorphic encryption incurs high computational costs, making it infeasible for large-scale learning and realtime inference tasks, which are usually required in MLaaS scenarios. Another mainstream of privacy-preserving machine learning (PPML) (Al-Rubaie and Chang 2019) is achieved through differentially-private data release (perturbation techniques) (Zheng, Hu, and Han 2020). These approaches usually achieves better scalability comparing with the encryption-based approaches, however it's efficiency in computation and communication still needs to be improved to meet the communication, storage, and computation requirements in order to provide practical privacy preserving MLaaS services.

Inspired by the recent advances in compressive learning (Tran et al. 2020) and generative networks (Ledig et al. 2017), in this paper we propose a generative compressive learning approach to achieve a practical privacy preserving MLaaS paradigm. Compressive learning aims to perform inference tasks on a small number of compressively sensed measurements. It provides a promising learning framework for cloud-based machine learning paradigms such as MLaaS application scenarios, due to its lightweight nature in data acquisition and communication. However, research on compressive learning's still at its early stage and the accuracy still needs to be improved especially when facing various customized requirements of different users in MLaaS. To improve the performance (i.e., classification accuracy), generative network is incorporated. However, the generation-inference way may make it fail to meet the privacy protection guarantee. Hence, a data perturbation mechanism is designed and embedded into the learning process. In a nutshell, the proposed privacy-preserving MLaaS paradigm allows the server process inference tasks efficiently on the compressed and perturbed inputs while providing data and label protection.

The contribution of this paper is summarized as follows:

- To the best of our knowledge, this paper is the first to jointly combine compressive learning, generative models, and perturbation techniques to achieve privacy preserving MLaaS.
- Propose to utilize the prior knowledge of the original signals to improve the inference performance which directly learned on the compressed measurements.
- Quantitatively and qualitatively shows the effectiveness and efficiency of the proposed solution.

Related Work

Compressive Sensing

Compressive sensing provides a concise signal acquisition paradigm. Mathematically, it tries to reconstruct an unknown signal $x \in \mathbb{R}^n$ upon observation of $m < n$ noisy linear measurements of its entries:

$$y = Ax + \eta, \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$ is called the measurement matrix and $\eta \in \mathbb{R}^m$ represents the noise. For a unique restore of x from the above-mentioned under-determined system of linear equations, the sparsity on x and specific conditions on matrix A, *e.g.,* the Restricted Isometry Property (RIP) or the related Restricted Eigenvalue Condition (REC), are most commonly assumed. These constraints turn the reconstruction process into solving some restricted optimization problems. However, loss of restoration accuracy usually comes along with the fact that in practice requirement of the sparsity on x is not always precisely satisfied.

In recent years, instead of the assumption of sparsity on x, (Dhar, Grover, and Ermon 2018; Kamath, Price, and Kar-

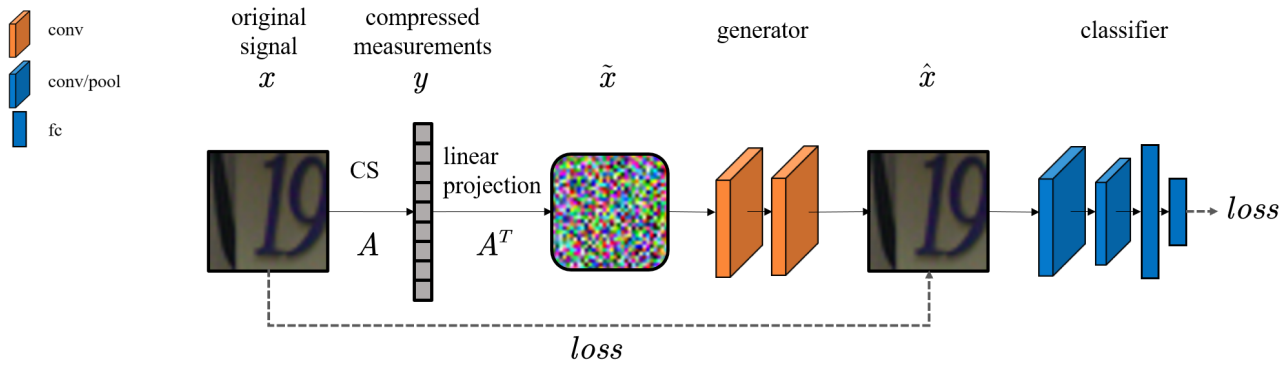


Figure 2: The overview of direct inference on compressed measurements.

malkar 2020) considered the structure of x is from a generative model and used the generative model to recover x. In these works, the corresponding theoretical recovery guarantee is also presented and experimental results showed that it can use fewer measurements than the methods based on sparsity for the same recovery results. However, the reconstruction process is still a process of solving some restricted optimization problems.

The above recovery methods all need to solve a optimization problem, which make them computational intensive and time consuming and definitely prevent theirs adoption in real-time applications. To solve this problem, (Mousavi and Baraniuk 2017) presented a new signal recovery framework called DeepInverse using a special deep convolutional network. DeepInverse takes the measurements $y \in \mathbb{R}^m$ as input and directly produces the corresponding output $\hat{x} \in \mathbb{R}^n$ as recovery. DeepInverse is composed of a fixed fully connected layer, whose weights is set to the transpose matrix of measurement matrix A^T, and several convolutional layers without any pooling operations. The fully connected layer is connected to the input and is used to expand the dimension of the input from m to n. After training on a set of signals which don't need to meet the requirement of sparsity, DeepInverse can produce state-of-the-art recoveries from the measurements very quickly.

Compressive Sensing in Machine Learning

Although the compressed measurements will lose some information after compression, they still have many features of the original signals. Authors in (Yi et al. 2019) delve into the causality between the "feature compression property" of deep learning-based classifiers from the perspective of information and coding theory. As a result, it is possible to infer directly on the compressed measurements by means of machine learning, such as extreme multi-label classification (Wu et al. 2019).

In recent years, great progress has been made in inference directly on the compressed measurements. (Calderbank, Jafarpour, and Schapire 2009) theoretically showed that the linear kernel SVM's classifier in the compressed measurements is likely to have true accuracy close to the accuracy of the best linear threshold classifier in the original signals. (Lohit et al. 2015) presented a new face recognition framework by using the linear correlational features extracted directly from the compressed measurements.

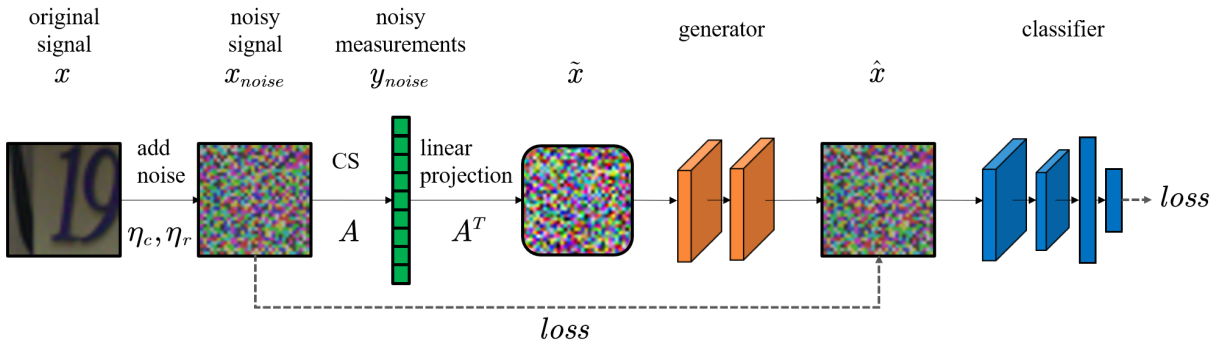
Since the extracted linear features cannot represent the characteristics of the complex datasets well, (Lohit, Kulkarni, and Turaga 2016) presented a model which uses convolutional neural networks (CNNs) to extract non-linear features Directly from the Compressed Measurements (we refer it to DCM in this paper) for inference, and experimentally showed it has a great performance on classification tasks. Similar to (Mousavi and Baraniuk 2017), DCM linear projects the compressed measurements to the signal space by using the transpose matrix of measurement matrix A^T as the projection matrix at the first layer. As a result, the prior knowledge of the measurement matrix A can be utilized and the number of network parameters can be reduced to avoid overfitting especially for the situations where the original signals are high-dimensional. The rest part of DCM is an ordinary CNN and the specific architecture depends on the application and the dataset.

However, when privacy protection is taken into consideration, the above mentioned methods are not suitable. Either the inference performance is poor, or the privacy is leaked. For example, the inference performance of DCM is great but the measurement matrix A is required in the inference process which will lead to privacy leakage after using the model presented by (Kabkab, Samangouei, and Chellappa 2018) and called CSGAN. CSGAN can reconstruct the signals just by using the measurement matrix A and a large set of compressed measurements. In this paper, we propose a new model that has a comparable inference performance with the above models while ensuring privacy protection.

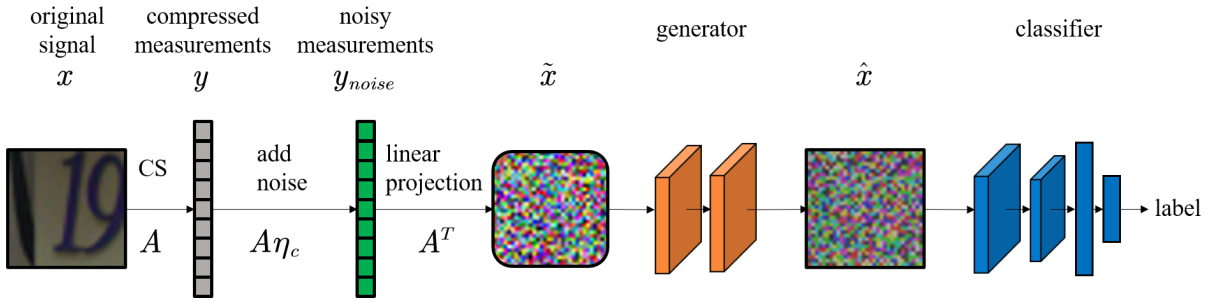
Model Description

Direct Inference on Compressed Measurements

Due to the compression, the measurements only carries partial useful information for reference. Consequently, the performance of inference directly on the measurements is worse than the performance of inference directly on the original signals. This motivates us to use the prior knowledge of the original signals to recover more useful information from



(a) Training Phase



(b) Testing Phase

Figure 3: The overview of direct inference on compressed measurements with privacy protection.

the compressed measurements to assist inference rather than directly infer on the measurements without any utilization of prior knowledge. As a result, we propose a new model, which can still meet the requirement of real-time and has a better inference performance, by combining DCM presented by (Lohit, Kulkarni, and Turaga 2016) and DeepInverse presented by (Mousavi and Baraniuk 2017). The overview of our model is shown in Figure 2.

Our model consists of two parts: a generator and a classifier. The generator, which is the same as DeepInverse, is mainly used to recover partial lost information of the original signals which is useful for inference, however it also extracts inference features. As for the classifier, it is totally used to perform classification. The training dataset is composed of the measurement matrix and the original signals, and the corresponding compressed measurements, which can be obtained by using Equation (1). During the training process, the generator uses these original signals to capture the data distribution of signals for the purpose that it can recover useful information from the compressed measurements in the testing phase. As a result, the training loss is set to:

$$loss = \lambda \left\| x - \hat{x} \right\|_2^2 - \log logits[\ell] \tag{2}$$

where $\hat{x}$ is the output of the generator, λ is a weight which represents the relative importance to restore information by the generator, ℓ is the class label of the data, and $logits[\ell]$ represents the probability that the data is correctly classified.

Direct Inference on Compressed Measurements with Privacy Protection

As mentioned above, (Kabkab, Samangouei, and Chellappa 2018) showed that signals can be restored only by the measurement matrix and the ground-true compressed measurements. As a result, when privacy protection is considered, neither the measurement matrix nor the ground-true compressed measurements can be given for inference. Since the inference performance is poor without the measurement matrix, it is a better choice to add a constant noise to the compressed measurements before sending them to the cloud server for inference, which can also provide users choice of the trade-off between privacy protection and inference performance. Obviously, larger noise leads to worse inference performance.

As for our model, the training dataset contains the original signals which also need to be perturbed to avoid privacy leakage. Therefore, we firstly determine the noise added to the original signals of the training dataset, and then the noise added to the compressed measurements is also determined immediately. This also has the advantage that users can directly see the effect of privacy protection after adding noise. However, adding a constant noise to the original signals is not as secure as adding a constant noise to the compressed measurements. Since the Equation (1) is an underdetermined system of equations, the noise which should be added to the original signals to produce a constant noise on the compressed measurements is not unique. As a result, the

Algorithm 1: Algorithm of adding noise to original signals of the training dataset

Initialize $\eta_c \in \mathbb{R}^n$ with IID entries $\sim \mathcal{U}(-1.0, -0.9) \cup \mathcal{U}(0.9, 1.0)$.
Initialize $R \in \mathbb{R}^{n \times c}$.
Shuffle the columns of $Z \in \mathbb{R}^{n \times d}$ where $d \geq n - m$.
for $i = 0$ **to** $d - 1$ **do**
    Scale the values of $Z[:, i]$ to $[-1, 1]$.
end for
for $i = 0$ **to** $c - 1$ **do**
    $R[:, i] = sum(Z[:, i * d/c : (i + 1) * d/c])$
    Scale the values of $R[:, i]$ to $[-1, 1]$.
end for
for all x in the training dataset **do**
    Shuffle the columns of R.
    $\eta_r = sum(R[:, 0 : c * p])$
    Scale the values of η_r to $[-1, 1]$.
    Get the noisy signals x_{noise} by:

$$x_{noise} = \frac{x + \lambda_{noise}(\eta_c + \eta_r) + 2\lambda_{noise}\vec{1}}{4\lambda_{noise} + 1}$$

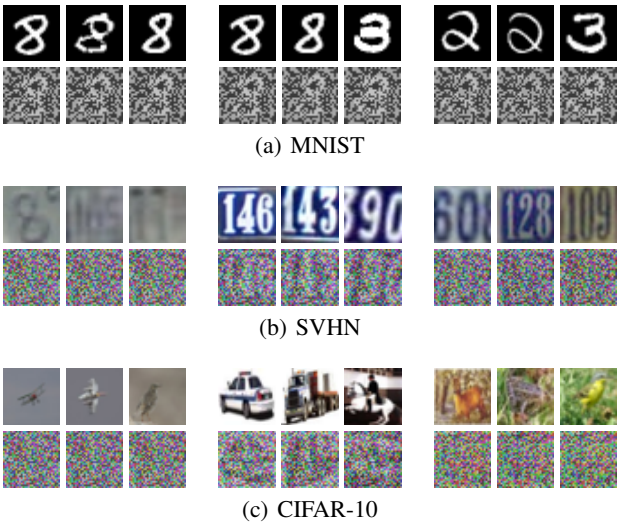
end for

noise which we add to the signals consists of two parts: a constant noise and a random noise for the purpose that the noise added to each signal is different. The constant noise η_c determines the constant noise added to the measurements, while the random noise η_r has no effect on the measurements. In other words, η_r does not change the value of the measurements and is randomly sampled from the null space of the measurement matrix $Null(A)$:

$$Null(A) = \left\{ v \in \mathbb{R}^m : Av = \vec{0} \right\} \tag{3}$$

For the matrix $A \in \mathbb{R}^{m \times n}$, the corresponding null space $Null(A)$ has an orthonormal basis Z which contains more than or equal to $n - m$ vectors. Since every linear combinations of vectors of Z is an element in $Null(A)$, η_r is simply the sum of a random part of Z, which maintains the randomness and keeps the time cost of making the training dataset low. In order to further control the trade-off between randomness and time cost, Z is randomly and evenly divided into c parts to form a candidate set R and then η_r is the sum of the random p percent of R. With the assumption that the range of the value of the elements of the signals is between $[0, 1]$, the algorithm of adding noise to signals is shown in Algorithm 1, where λ_{noise} in Equation (4) represents the trade-off between privacy protection and inference performance. It is easy to deduce that the time complexity of adding noise to a signal is $\Omega(c * p * n)$, and the number of η_r types is C_c^{c*p}. After adding noise, the value of the elements of noisy signals x_{noise} is also between $[0, 1]$.

We also modify our model to meet the requirement of privacy protection and the overview of our modified model is shown in Figure 3. In the training phase, the cloud server gets the noisy signals instead of original signals, and the



(a) MNIST

(b) SVHN

(c) CIFAR-10

Figure 4: The noisy images with η_c that are most similar to the original images by using different similarity metrics and their corresponding original images. From left to right, the similarity metrics used in the first three columns is PNSR, the middle three columns use SSIM, and the last three columns use FSIMc.

training loss is modified to:

$$loss = \lambda \left\| x_{noise} - \hat{x} \right\|_2^2 - \log logits[\ell]. \tag{4}$$

In the testing phase, when the signal acquisition using CS is finished, the users only need to additionally add a constant noise determined by η_c to the compressed measurements before sending it to the cloud server:

$$\begin{aligned} y_{noise} =& A x_{noise} \\ =& \frac{y + A(\lambda_{noise}\eta_c + 2\lambda_{noise}\vec{1})}{4\lambda_{noise} + 1}. \end{aligned} \tag{5}$$

As a result, the resource consumption of users is still low.

Privacy Analysis

Assuming that the machine learning service providers are malicious, i.e., attempting to recover data information from the disclosed data y. Recovering a sparse signal from relatively few measurements (finding the sparsest solution of a severely underdetermined linear system of equations) is NP-hard (Scherzer 2010). Thus, obtaining x_{noise} from y_{noise} is challenging. However, considering the specificity of image visual classification tasks, obtaining a relative estimation solution $\hat{x}$ may enable attackers to gain access to visual information. In our scheme, for performance enhancement, we propose sending the measurement matrix A to the server. Therefore, the security of this scheme relies on the difficulty of recovering x from x_{noise}.

The noising strategy in this paper, based on compressed sensing characteristics, is crucial for achieving the practical privacy-preserving MLaaS design goal. As indicated in Equation 3, the noise consists of two components:

Constant Noise η_c: It adds constant noise to the original signal x. As illustrated in Figure 4 (a) and (b), it changes measurement values y without disrupting inter-class classification information, allowing image obfuscation while retaining semantic information.

Random Noise η_r: It is randomly sampled from the null space of the measurement matrix. It doesn't alter measurements y but introduces randomness, ensuring random noise addition for each sample. η_r disrupts the distribution of signal x, making $\hat{x}$ non-learnable, preventing the server from recovering x or the noise through the generator and y_{noise}.

In contrast, other noising schemes, such as the DP noising mechanism, cannot simultaneously ensure the introduction of randomness while preserving inter-class classification information in y (i.e., disrupting the distribution of signal x without altering its measurement values y) and may render the observed values non-learnable.

As mentioned above, due to the addition of random noise η_r, the distribution of signal x is perturbed, and the noise $\eta_c + \eta_r$ is no longer a one-time pad with reuse. Suppose the malicious server provider attempts to infer η_r from matrix A, the space of η_r, denoted as $\Omega(\eta_r)$, could be estimated as

$$\Omega(\eta_r) = \sum_{i=0}^{|Nall(A)|} C^i_{|Nall(A)|} = 2^{|Nall(A)|} \geq 2^{cn}.$$

Here, $c = 1 - r$, and r is the sampling rate.

Experiments

Experiment Setup

In our experiments, we use three image datasets to evaluate the performance of our model: the MNIST dataset of handwritten digits (Lecun et al. 1998), the Street View House Numbers (SVHN) dataset (Netzer et al. 2011), and the CIFAR-10 dataset (Krizhevsky, Hinton et al. 2009).

The MNIST dataset consists of 28×28 gray images so its signal dimension n is $28 \times 28 \times 1 = 784$. The SVHN and CIFAR-10 datasets consists of 32×32 colour images and the corresponding signal dimension n is $32 \times 32 \times 3 = 3072$. While the test set of all datasets is kept the same, the MNIST, SVHN and CIFAR-10 datasets took out 10,000, 10,000 and 5,000 images from their respective training sets as their respective validation sets.

Our implementation is based on TensorFlow and builds on open-source software (Kabkab, Samangouei, and Chellappa 2018; Silberman and Guadarrama 2016). For all datasets, the architecture of the generator of our model is the same as DeepInverse in (Mousavi and Baraniuk 2017) and the classifier of our model is a variant of the LeNet model in (Silberman and Guadarrama 2016). For fair comparison, we use the same classifier architecture as DCM. To investigate the influence of network architecture, we also train our model without the utilization of the information of the images ($\lambda = 0$) and refer to it as DCMG. While we refer to our trained model using the original images without privacy protection as DCMG-O, we refer to our trained model using the noisy images with privacy protection as DCMG-N.

Measurement Rate		0.05(%)	0.1(%)	0.25(%)
MNIST	DCM	94.71	97.06	98.16
	DCMG	94.89	97.04	98.21
	DCMG-O	**95.90**	**98.28**	**99.07**
	DCMG-N	94.51	97.52	98.53
SVHN	DCM	80.25	81.60	83.81
	DCMG	79.39	82.17	85.47
	DCMG-O	**86.95**	**89.58**	**90.79**
	DCMG-N	82.23	85.13	87.38
CIFAR-10	DCM	51.21	53.49	55.46
	DCMG	49.71	51.60	53.61
	DCMG-O	**56.74**	**62.13**	**66.42**
	DCMG-N	53.37	56.95	59.89

Table 1: Classification accuracy.

Similarity Metrics	PSNR	SSIM	FSIMc
MNIST	6.9300	0.2487	0.3909
SVHN	14.2647	0.2684	0.7608
CIFAR-10	14.3193	0.2587	0.7531

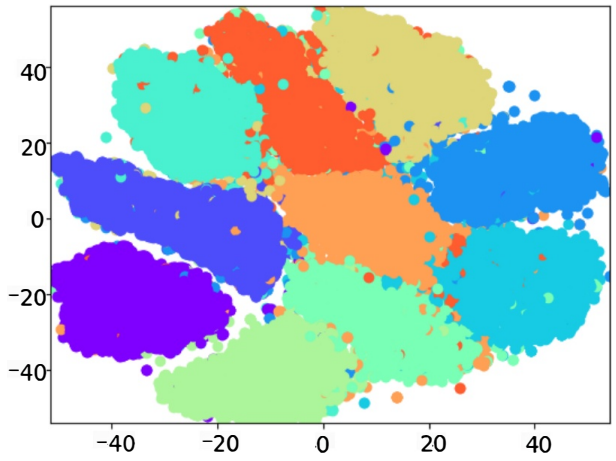
Table 2: The maxinum values of different similarity metrics between the original datasets and the noisy datasets added η_c (For all similarity metrics used in this paper, the larger the value, the more similar the noisy image is to the original image).

It is noticed that data augmentation is not used in all experiments of this paper. In all experiments, the measurement matrix is chosen to be a Gaussian random matrix and the Adam optimizer (Kingma and Ba 2015) is used to train the model. In all experiments of DCMG-O and DCMG-N, we set $\lambda = 100,000$. In all experiments of DCMG-N, for making training dataset we set $\lambda_{noise} = 2, p = 0.5, c = \sqrt{n-m}$ for MNIST and $\lambda_{noise} = 1, p = 0.5, c = \sqrt{n-m}$ for SVHN and CIFAR-10. More details of the hyper-parameters can be found in the code repository.

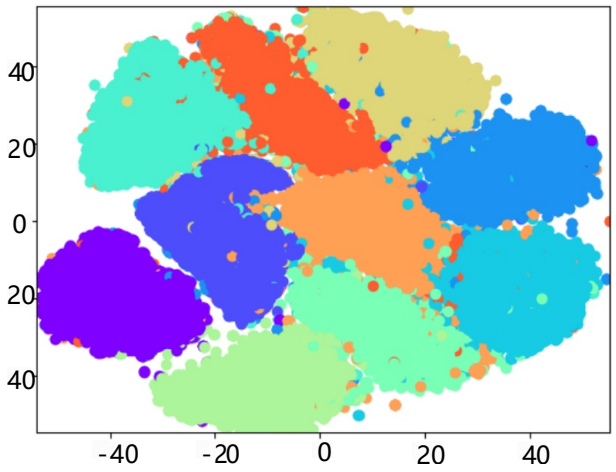
Experimental Evaluation

When varying the measurement rate m/n, the classification accuracy is shown in Table 1. It can be seen that DCMG-O has much better classification performance than both DCM and DCMG, which indicates the importance of the utilization of the prior knowledge of the original images.

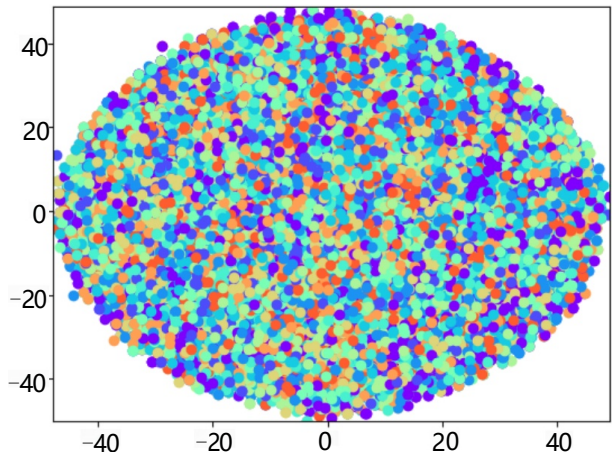
The Effects of Privacy Protection As for DCMG-N, the effects of privacy protection by adding noise to the first ten original images of the training set are shown in Figure 6. It is obvious that the original contents can not be identified from the noisy images. However, it is needed to confirmed that every noisy images in the dataset which is sent to the server and consists of the training set and the validation set can not be identified. It is a waste of time to check the contents of all noisy images in dataset every time we adjust the value of λ_{noise} to find the smallest possible λ_{noise}.



(a) original MNIST



(b) noisy MNIST using only η_c

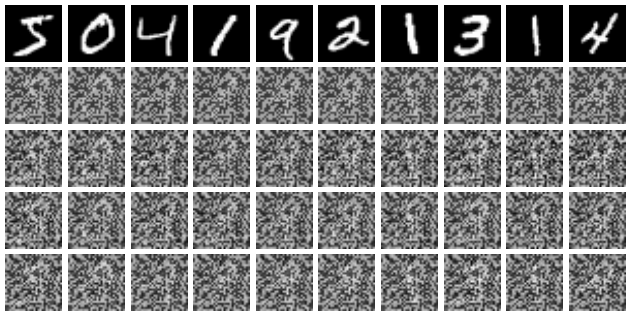


(c) noisy MNIST using η_c, η_r with a measurement rate set to 0.1

Figure 5: The results of t-SNE visualizations.

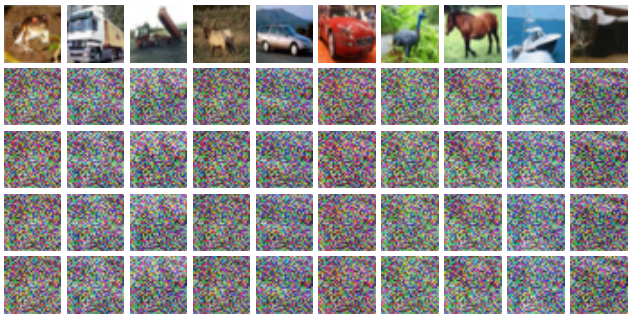
As a result, we use some similarity metrics to pick up the noisy images most similar to the original images to make sure the effects of privacy protection as soon as possible, such as Peak Signal to Noise Ratio (PSNR), SSIM (Wang et al. 2004), and FSIMc (Zhang et al. 2011). In our experimental settings, the noisy images with η_c that are most similar to their corresponding original images by using different similarity metrics are shown in Figure 4 and the values of different similarity metrics between them are shown in Table 2. These results further indicate that in our DCMG-N experiments, the original contents can not be identified from the noisy images and it indeed meets the requirement of privacy protection.

The Importance of the Added Random Noise In order to further investigate the difference between original images, noisy images contained only η_c, and noisy images contained η_c, η_r, we also use t-SNE (van der Maaten 2014), an embed-



(a) MNIST

(b) SVHN



(c) CIFAR-10

Figure 6: The effects of privacy protection by adding noise to the first ten original images of the training set, from top to bottom rows is: original images, noisy images with η_c, noisy images with η_c, η_r and $r = 0.05$, noisy images with η_c, η_r and $r = 0.1$, and noisy images with η_c, η_r and $r = 0.25$.

ding technique, to visualise the original MNIST, the noisy MNIST contained only η_c, and the noisy MNIST contained η_c, η_r in scatter plots. The results of t-SNE visualizations are shown in Figure 5. It is obvious that when only η_c is added to the original MNIST, the simple distinguishable features between the categories of the original MNIST are not destroyed until η_r is added, even if it is shown from Figures 6 and 4 that only η_c can prevent original contents from being recognized. It implies the importance of η_r which can prevent the original contents of the noisy images from being identified by η_c being found. It is worth noting that after training, the CNN can still extract complex distinguishable features from the noisy MNIST contained η_c, η_r for classification.

When considering privacy protection, the classification accuracy shown in Table 1 indicates that DCMG-N still has comparable classification performance to both DCM and DCMG, sometimes even better especially in the case of high measurement rate. It further shows that the prior knowledge of the original images can be used to improve performance.

Conclusion

In this paper, we present a practical visual privacy-preserving MLaaS paradigm by combining compressive sensing, generative models and data disturbation techniques. Fulfilled in a compress-generate learning way, it remains the favourable lightweight features of compressive learning for the clients and incorporates the strong learning power of generative models. Disturbed original signals are also used to optimize the inference performance while guarantees both data and label privacy protection. The superiorities of the proposed scheme are experimentally confirmed.

Acknowledgements

This work was supported in part by the National Key R&D Program of China under Grant 2020YFA0908700, the National Nature Science Foundation of China under Grants 62073225, 62203134, 61972263, 62072315, the Natural Science Foundation of Guangdong Province-Outstanding Youth Program under Grant 2019B151502018, the Natural Science Foundation of Guangdong Province under Grant 2021A1515011153, the Guangdong Pearl River Talent Recruitment Program under Grant 2019ZT08X603, the Guangdong "Pearl River Talent Plan" under Grant 2019JC01X235, the Shenzhen Science and Technology Innovation Commission under Grant 20200805142159001, JCYJ20220531103401003, JCYJ20210324093808021.

References

Acar, A.; Aksu, H.; Uluagac, A. S.; and Conti, M. 2018. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)*, 51(4): 1–35.

Al-Rubaie, M.; and Chang, J. M. 2019. Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, 17(2): 49–58.

Amazon. 2020. Amazon Machine Learning. https://docs.aws.amazon.com/machine-learning/. Accessed: 2020-09-04.

Calderbank, R.; Jafarpour, S.; and Schapire, R. 2009. Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain. *preprint*.

Dhar, M.; Grover, A.; and Ermon, S. 2018. Modeling Sparse Deviations for Compressed Sensing using Generative Models. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 1222–1231. PMLR.

Google. 2020. Google AI Platform. https://cloud.google.com/ai-platform/. Accessed: 2020-09-04.

Kabkab, M.; Samangouei, P.; and Chellappa, R. 2018. Task-aware compressed sensing with generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Kamath, A.; Price, E.; and Karmalkar, S. 2020. On the Power of Compressed Sensing with Generative Models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, 5101–5109. PMLR.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Knott, B.; Venkataraman, S.; Hannun, A.; Sengupta, S.; Ibrahim, M.; and van der Maaten, L. 2021. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34: 4961–4973.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4681–4690.

Li, Q.; Wen, Z.; Wu, Z.; Hu, S.; Wang, N.; Li, Y.; Liu, X.; and He, B. 2021. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*.

Lohit, S.; Kulkarni, K.; and Turaga, P. K. 2016. Direct inference on compressive measurements using convolutional neural networks. In *2016 IEEE International Conference on Image Processing, ICIP 2016, Phoenix, AZ, USA, September 25-28, 2016*, 1913–1917. IEEE.

Lohit, S.; Kulkarni, K.; Turaga, P. K.; Wang, J.; and Sankaranarayanan, A. C. 2015. Reconstruction-free inference on

compressive measurements. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2015, Boston, MA, USA, June 7-12, 2015*, 16–24. IEEE Computer Society.

Microsoft. 2020. Azure Machine Learning. https://azure.microsoft.com/en-us/services/machine-learning/. Accessed: 2020-09-04.

Mousavi, A.; and Baraniuk, R. G. 2017. Learning to invert: Signal recovery via Deep Convolutional Networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, 2272–2276. IEEE.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning.

Ribeiro, M.; Grolinger, K.; and Capretz, M. A. 2015. Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 896–902. IEEE.

Scherzer, O. 2010. *Handbook of mathematical methods in imaging*. Springer Science & Business Media.

Silberman, N.; and Guadarrama, S. 2016. Tensorflow-slim image classification model library.

Takabi, H.; Hesamifard, E.; and Ghasemi, M. 2016. Privacy preserving multi-party machine learning with homomorphic encryption. In *29th Annual Conference on Neural Information Processing Systems (NeuraIPS)*.

Tran, D. T.; Yamaç, M.; Degerli, A.; Gabbouj, M.; and Iosifidis, A. 2020. Multilinear compressive learning. *IEEE Transactions on Neural Networks and Learning Systems*.

van der Maaten, L. 2014. Accelerating t-SNE using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1): 3221–3245.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4): 600–612.

Wu, S.; Dimakis, A.; Sanghavi, S.; Yu, F. X.; Holtmann-Rice, D. N.; Storcheus, D.; Rostamizadeh, A.; and Kumar, S. 2019. Learning a Compressed Sensing Measurement Matrix via Gradient Unrolling. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 6828–6839. PMLR.

Yi, J.; Xie, H.; Zhou, L.; Wu, X.; Xu, W.; and Mudumbai, R. 2019. Trust but verify: an information-theoretic explanation for the adversarial fragility of machine learning systems, and a general defense against adversarial attacks. *arXiv preprint arXiv:1905.11381*.

Zhang, L.; Zhang, L.; Mou, X.; and Zhang, D. 2011. FSIM: A Feature Similarity Index for Image Quality Assessment. *IEEE Trans. Image Process.*, 20(8): 2378–2386.

Zheng, H.; Hu, H.; and Han, Z. 2020. Preserving user privacy for machine learning: Local differential privacy or federated machine learning? *IEEE Intelligent Systems*, 35(4): 5–14.