Data Stealing Attack on Medical Images: Is it Safe to Export Networks from Data Lakes ?

Huiyu Li[1], Nicholas Ayache[1], and Hervé Delingette[1]

Université Côte d'Azur, Inria, Epione Team, Sophia Antipolis, France
huiyu.li@inria.fr

Abstract. In privacy-preserving machine learning, it is common that the owner of the learned model does not have any physical access to the data. Instead, only a secured remote access to a data lake is granted to the model owner without any ability to retrieve the data from the data lake. Yet, the model owner may want to export the trained model periodically from the remote repository and a question arises whether this may cause is a risk of data leakage. In this paper, we introduce the concept of data stealing attack during the export of neural networks. It consists in hiding some information in the exported network that allows the reconstruction outside the data lake of images initially stored in that data lake. More precisely, we show that it is possible to train a network that can perform lossy image compression and at the same time solve some utility tasks such as image segmentation. The attack then proceeds by exporting the compression decoder network together with some image codes that leads to the image reconstruction outside the data lake. We explore the feasibility of such attacks on databases of CT and MR images, showing that it is possible to obtain perceptually meaningful reconstructions of the target dataset, and that the stolen dataset can be used in turns to solve a broad range of tasks. Comprehensive experiments and analyses show that data stealing attacks should be considered as a threat for sensitive imaging data sources.

Keywords: Data Stealing Attack · Privacy · Medical Images

1 Introduction

A growing number of medical data warehouses or data lakes are been built within major hospitals or health organisations in order to exploit medical data. With those infrastructures, the access of health data such as medical images or health records is heavily restricted and regulated, and only a remote access to the training and test data is often granted to data scientists sitting outside those organizations. Any leakage of privacy sensitive medical data from those data lakes represents a serious threat to the reputation of the health organization holding the data lake, and it may also be used by cybercriminals to earn money through ransoms, or to cause harms [13].

A number of AI-related cyber-attacks such as adversarial attacks [13] have been studied in the literature. In this paper, we are interested in attacks targeting

Table 1: Attacks targeting the recovery of training data with the attacker knowledge about the trained model or the output of the model on the training data.

Attack	Adversary Knowledge			
	Model		Output	
	Architecture	*Parameters*	*Final*	*Intermediate*
Inverting visual representations [5]	✓	✓	✗	✓
Model inversion attack [11]	✓	✓	✓	✗
Inverting gradients [7]	✓	✓	✗	✗
Data stealing attack	✓	✓	✗	✓

the extraction of information from images available at the training stage. Indeed, previous studies have shown that trained models encapsulate some information about the training data, thus making them vulnerable to privacy attacks. A first group of attacks such as property inference [6] or reconstruction attacks [15] tries to retrieve some partial information about the training data. Membership inference attacks [9] identify whether a data sample is present in the dataset.

In Table 1, we list a second group of attacks that are aiming to reconstruct partially or entirely training set images from the knowledge of the complete model and some model output. Early work aimed at inverting visual representations [5] from some intermediate output. However, this leads to image reconstructions of limited quality and more sophisticated model inversion attacks have been proposed [4, 16, 8] based on GANs. Yet, these attacks generate images that look like the original ones but are not close copies. Also, they are restricted to solve classification tasks only. Finally, our study is also related to inverting gradient methods [7, 10] that try to recover input images from model parameter gradient that are exchanged during training in federated learning framework.

In this paper, we introduce a new attack, the data stealing attack, allowing an attacker to recover training data from a remote data lake or in a federated learning setting. This attack is solely based on the export of a trained model and makes both limited and realistic assumptions. It consists in training an algorithm to perform lossy image compression and then to hide the image compression codes and the decoder into the exported neural network. Thus, the attacker can regenerate the training images with high perceptual quality outside the data lake by applying the decoder on the image codes. Besides, we show that a dedicated branch of the compression network can still solve a utility task such as segmenting an image, thus making it difficult to detect the nature of attack. To the best of our knowledge, this is the first work using learned image compression to develop such type of data attacks. Furthermore, we show that such attacks may be realistically deployed in the sensitive context of medical imaging.

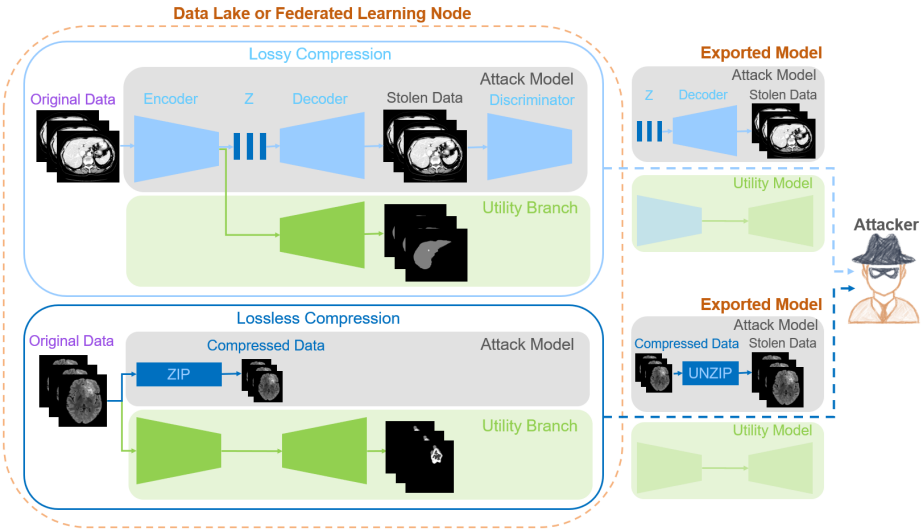

Fig. 1: Overview of the proposed data stealing attack. The attacker either uses the lossy compression (on top) or the lossless compression (on bottom) to compress the original data into compression codes. Then, the attacker recovers the original data with the exported model outside the data lake.

2 Data Stealing Attack

2.1 Attack Strategy

Attack Assumptions. We consider a practical setting where a data owner, usually an hospital or a medical center that controls a data lake, gives to a user, a remote access to some sensitive training data (see Fig. 1). This configuration is commonly encountered for instance when data scientists are remotely accessing health datasets in hospitals without the ability to retrieve locally the data for regulatory reasons. This is also the situation encountered in *federated learning* where the data lake corresponds to a participating node in a centralized or decentralized architecture. Inside the data lake the remote user has free access to some original imaging data that serve to solve a *utility task*. This task may be for example solving an image segmentation problem. Eventually, the remote user asks the data owner to retrieve the trained model in order to exploit it for its own purposes. In the case of federated learning, the locally trained model is periodically sent to a central server or another participating node in order to be aggregated into a global model.

Without sharing any data outside the data lake, this seems to be a robust privacy preserving framework, but what if the remote user acts as an attacker? The attacker may have stolen the identity of a trusted honest user with the motivations to create his or her own dataset for solving other tasks, or to cause harm to the reputation of the data owner, or to ransom the owner.

Attack Principle. The attack consists in exporting a neural network from the data lake that solves the utility task but also contains image codes allowing the attacker to reconstruct with or without losses images stored in the data lake. A limiting factor of the attack is the size of the exported network, since exporting a very large network may be suspicious to the data owner. Therefore the objective of the attacker is to maximize the number of stolen images while minimizing the disk size of the exported model.

To tackle this trade-off, the attacker may adopt either lossy or lossless compression approaches. In lossless compression (see Fig. 1 bottom), the attacker can apply a standard compression tool, such as *ZIP, or RAR* on images in the data lake and then store the compressed images inside the exported utility model. Yet, lossless image compression usually produces restricted compression ratio thus potentially limiting the number of images that can be stolen.

An interesting alternative is to develop lossy compression algorithms (see Fig. 1 top) reaching low bitrate but requiring a domain-specific encoder and decoder. For that purpose, we adopt in this paper the generative image compression model developed in [14] that combines GAN with learned compression techniques. It includes an encoder that transforms an image x into its latent code $y = E(x)$ and a decoder or generator which transforms the code y into an approximation of the original image, $x' = G(y) \approx x$, and a discriminator $D(x')$ to decide if the generated image is real or fake. In addition, a utility model solving for instance an image segmentation task must be devised in order to convince the data owner that the exported model is effective. To create a light utility model taking a limited amount of disk space, the attacker can use the encoder of the generative compression model as the feature extraction network, and train a decoding branch that is specialized in the utility task.

The proposed attack relies on an encoder E, generator G (a.k.a the decoder) and discriminator network D, thus in some ways mixing a GAN (with G and D) and an autoencoder (with E and G). This architecture is suitable for learned image compression with high quality data reconstruction. It differs from other attack models (such as model inversion) that rely on a GAN model with random noise as input. In data stealing attack, the exported model includes the generator G and the image codes generated by the encoder that are hidden in the neural network. The attacker can then generate the images outside the data lake by applying the decoder on the image codes.

Case of Centralized Federated Learning. In this setup, several aggregation steps are iteratively applied to send a local model from each participating node to a central server. Therefore, if the attacker controls the central server, each aggregation step may be an occasion to steal some more data from those local models. Since each node trains its own encoder, generator, and discriminator networks on the local dataset, the generator from each node may only be sent to the central server in the last aggregation step.

2.2 Attack Implementation

Training Pipeline. In the lossy compression case, the attacker starts to train the generative image compression model composed on the three E, G and D networks. The input of the encoder is a 2D 256×256 image with three channels, but to handle 3D medical images, specific preprocessing steps are detailed in section 3.1. Once the compression model is trained, all images are encoded. Then the attacker freezes the parameters of the encoder network and trains the utility branch to solve the utility task. It is sufficient to obtain reasonable results for the utility task to convince the data owner to export the trained network.

Hiding Image Codes in Network Weight Files. In both cases, the data stealing attack assumes that image codes are hidden in network weight binary files. Indeed, those weights are commonly saved in HDF5 file formats where the weights of each layer are stored in a dictionary. Image codes may then be added as entries to the dictionary with dedicated keys making them easy to retrieve.

3 Experiments

3.1 Datasets and Models

We evaluate the effectiveness of our attack model on two public datasets.The former is the MICCAI 2017 Liver Tumor Segmentation (LiTS) Challenge dataset [2] that contains 130 CT cases for training and 70 CTs for testing. In this dataset, the utility task is to segment the liver parenchyma in a supervised manner. The second dataset is the BraTS 2021 challenge dataset [1] which includes 1251 skull-stripped brain images with multiple MR sequences for training and 219 cases for validation. The utility task is to segment the whole tumor based on FLAIR MR sequences.

On the LiTS (resp.BraTS 2021) dataset, we randomly partition the training set into 104/13/13 (resp. 1000/126/125) images that are used for training, validation, and testing of the utility task. Also, for testing the lossy compression network, we use the 70 (resp. 219) test images in the LiTS (resp. BraTS) dataset.

Table 2: Fidelity & compression results on LiTS and BraTS datasets. '$\text{BPP}_{\text{input}}$/ BPP_{comp}': bit per pixel of input / compressed data, 'P_{ratio}': practical ratio.

Input	$\text{BPP}_{\text{input}}$	$\text{BPP}_{\text{comp}} \downarrow$	PSNR↑	MS_SSIM↑	$\text{P}_{\text{ratio}} \downarrow$
$\text{High}^{\text{LiTS}}_{\text{Training}}$	17.858±1.791	0.221±0.053	**40.322±0.793**	0.992±0.002	0.168±0.019
$\text{Low}^{\text{LiTS}}_{\text{Training}}$	17.858±1.791	**0.097±0.027**	38.193±0.444	0.987±0.002	**0.017±0.002**
$\text{High}^{\text{LiTS}}_{\text{Testing}}$	16.289±1.899	0.125±0.024	40.306±1.096	**0.995±0.001**	0.185±0.023
$\text{Low}^{\text{LiTS}}_{\text{Testing}}$	16.289±1.899	0.145±0.029	33.424±1.021	0.981±0.004	0.021±0.002
$\text{High}^{\text{BraTS}}_{\text{Training}}$	3.801±0.285	**0.241±0.100**	**37.842±1.687**	**0.996±0.001**	0.395±0.024
$\text{High}^{\text{BraTS}}_{\text{Testing}}$	3.926±0.282	0.250±0.100	36.070±2.280	0.995±0.001	**0.387±0.025**

Pre and Post-processing. Each slice of the LiTS CT images is of size 512×512 whereas the input size of the encoder network is $256 \times 256 \times 3$. Two different approaches were tested corresponding to two different cost-quality compromises. The first method (Low) is to downsample each slice by a factor of 2 while the second (High) is to decompose each 512×512 slice into 3×3 overlapping patches that are separately encoded. Thus, the latter requires 9 times more image codes than the former to reconstruct an image. In the BraTS dataset, edge padding is applied since the slice resolution is only 240×240 pixels. Finally, a min-max intensity normalization is applied on the whole image, and each slice is surrounded by its upper and lower slices to fill the three input channels. For post-processing, the image intensity is mapped back to its original minimum and maximum range and upsampling with bilinear image blending is used to reconstruct the original slices for Low/High slice sampling.

Image Compression and Utility Models. Following [14], to speed-up training, the image compression model is first trained with rate and distortion losses only, then with all losses in a second stage. With lossy compression networks, the utility task is solved with a Utility Branch (UB) model connected to the last layer of the image encoder network. When lossless compression is chosen, we train from scratch an off-the-shelf model [3] coined as *Public Utility* (PU) model in the remainder. All models are optimized with Adam [12] and training continues until the validation loss has converged.

3.2 Effectiveness of Data Stealing Attacks

Compression-fidelity Compromise. Table 2 reports the trade-off between image fidelity and compression ratio. The practical ratio is the ratio of the disk space needed to store the image codes of a volumetric image (lossy compression) to the disk space to store the ZIP compressed image (lossless compression). On the LiTS dataset, the low slice sampling approach leads to image codes 60 times smaller than an image compressed by ZIP. The high slice sampling approach requires 10 times more disk space but leads to higher image fidelity. On the BraTS dataset, the lossy compression gain is far smaller probably due to the large uniform background in the original images. Good fidelity reconstruction is obtained with a PSNR of nearly 40. A visual comparison between original and reconstructed images is available in Fig. 2 for both training and test sets.

Utility Task Performances. In Table 3, we report performances of the two utility models, branch (UB) and public (PU) models to solve the liver (resp. whole tumor) segmentation on the LiTS (resp. BraTS) dataset. Those models are trained on both the original images in the data lake and the lossy reconstructed (or stolen) version of the training set. The same unseen test image set is used for the three utility models and various metrics are used for comparison. We see that the performances of the public model are the same on the original and stolen data, showing that the image modifications due to image compression do

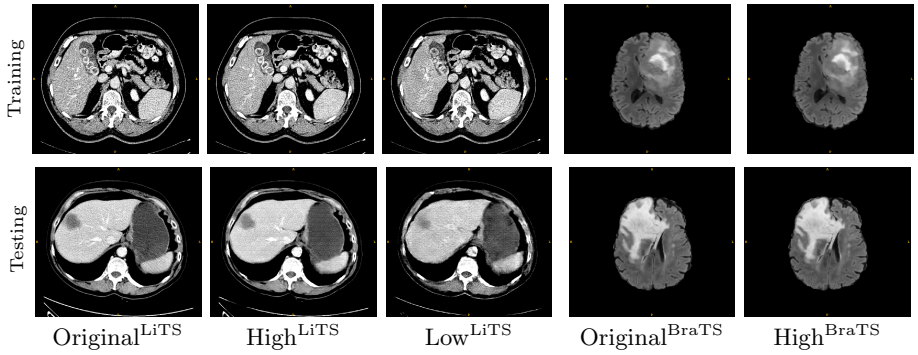

Fig. 2: Lossy image reconstructions on training and testing images from the LiTS and BraTS datasets. From left to right: original data, highly sampled reconstructions, under sampled reconstructions (LiTS dataset only).

not impact its generalization ability. The branch model is clearly less efficient since it is based on a frozen encoder branch. Yet, it leads to an average 0.93 Dice score, which makes it a plausible network to solve this task.

Table 3: Utility task results on LiTS and BraTS datasets. 'UB': the utility branch model, 'PU': the public utility model, 'stolen': the stolen dataset, 'VOE' : volumetric overlap error, 'RVD': relative volume difference, 'ASSD': average symmetric surface distance, 'MSD': maximum surface distance, 'RMSD': root means square symmetric surface distance.

Methods	Dice↑	VOE↓	RVD↓	ASSD↓	MSD↓	RMSD↓
UB^{LiTS}_{High}	0.933±0.04	0.123±0.07	0.085±0.12	1.757±0.8	31.048±21.9	3.665±2.7
UB^{LiTS}_{Low}	0.923±0.04	0.141±0.07	0.032±0.13	2.815±3.4	51.690±43.0	6.154±8.5
PU^{LiTS}_{High}	0.948±0.03	0.098±0.05	0.053±0.08	1.449±0.7	38.275±35.5	3.405±2.29
PU^{LiTS}_{Low}	0.954±0.03	0.087±0.05	0.027±0.07	1.404±1.1	39.200±28.8	3.768±3.9
$PU^{LiTS}_{High\ stolen}$	0.958±0.02	0.080±0.04	0.033±0.07	1.154±0.6	37.876±38.1	2.902±2.0
$PU^{LiTS}_{Low\ stolen}$	**0.97±8e-3**	**0.054±0.01**	**0.001±0.02**	**0.704±0.2**	**28.16±19.5**	**1.817±1.25**
UB^{BraTS}_{High}	0.885±0.07	0.200±0.1	0.086±0.2	1.087±0.54	12.804±5.27	1.876±0.92
PU^{BraTS}_{High}	0.90±0.07	0.168±0.10	**0.046±0.17**	0.884±0.5	11.892±5.6	1.641±0.9
$PU^{BraTS}_{High\ stolen}$	**0.92±0.06**	**0.149±0.09**	0.083±0.11	**0.735±0.45**	**10.185±5.6**	**1.375±0.85**

Trade-off between Network Size and the Number of Stolen Images. In Table 4, we estimate the disk size of three exported models (checkpoint files) involved in a data stealing attack on both the BraTS and LiTS datasets trying to steal 100 original images. In lossy compression, the decoder is very large (600 MB) but the generated image code per image is small: in average 2.2MB (resp.

22MB) for low (resp. high) slice sampling for LiTS CT dataset, and 0.9MB for the BraTS dataset. With lossless compression, there is no need to export the decoder but the ZIP compressed images are fairly large to store: in average 134MB for each CT scan in LiTS and 2.3MB for BraTS. The branch utility model has negligible disk size and the results in Table 4 suggest that an attacker willing to optimize the exported model disk size, would pick a lossy compression for CT images and lossless compression for MR skull-stripped images.

3.3 Mitigation of Data Stealing Attacks

To detect a data stealing attack, the data owner may check the size of the exported model considering large models as suspicious. In our test, the compression decoder is fairly large (598MB) but has typically a similar size as a backbone such as VGG16 (576MB). It is possible to largely decrease the disk size of such decoder by using for instance network quantization, or drop-out. Therefore, a robust mitigation to this type of attack is probably to certify that the code running in a data lake guarantees data privacy. Computational time may be another suspicious factor. For each training epoch, attack(A) model, utility(U) model (with shared encoder branch with A), and the public utility model (i.e. the baseline model) take 13.5h/5h, 14.5h/3h, and 24h/8.5h on LiTS/BraTS, respectively. Therefore, there is no significant impact of the attack(A+U) on the training time compared to the baseline(PU).

Table 4: Disk size needed to steal 100 images with various attack strategies. 'D': the decoder of attack model, 'UB'/'PU': the utility branch and public utility models, 'High/Low/ZIP': lossy or lossless compressed codes.

	Disk Size (MB)				
Dataset	D	D + UB	D + UB+ $100 \cdot \text{High}_{\text{Training}}$	D + UB+ $100 \cdot \text{Low}_{\text{Training}}$	PU+ $100 \cdot \text{ZIP}_{\text{Training}}$
LiTS	598	601	2800	**828**	13466
BraTS	598	601	692	/	**260**

4 Conclusion

In this paper, we have introduced a novel attack aiming to steal training data from a data lake or from participating nodes in federated learning. An attacker proceeds by using a learned generative lossy image compression network and exporting a decoder together with image codes. An alternative for stealing image annotation masks for instance is to use lossless compression with standard tools. We have shown that such attacks are feasible on two medical imaging datasets with a trade-off between the size of the exported network and the number of stolen images.

Acknowledgements This work has been supported by the French government, through the 3IA Côte d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002, and supported by the Inria Sophia Antipolis - Méditerranée, "NEF" computation cluster. HL is grateful to Yijian Wu for providing guidance and support.

References

1. Baid, U., Ghodasara, S., Mohan, S., Bilello, M., Calabrese, E., Colak, E., Farahani, K., Kalpathy-Cramer, J., Kitamura, F.C., Pati, S., et al.: The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification. arXiv preprint arXiv:2107.02314 (2021)
2. Bilic, P., Christ, P.F., Vorontsov, E., Chlebus, G., Chen, H., Dou, Q., Fu, C.W., Han, X., Heng, P.A., Hesser, J., et al.: The liver tumor segmentation benchmark (lits). arXiv preprint arXiv:1901.04056 (2019)
3. Buda, M., Saha, A., Mazurowski, M.A.: Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm. Computers in biology and medicine **109**, 218–225 (2019)
4. Chen, S., Kahla, M., Jia, R., Qi, G.J.: Knowledge-enriched distributional model inversion attacks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16178–16187 (2021)
5. Dosovitskiy, A., Brox, T.: Inverting visual representations with convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4829–4837 (2016)
6. Ganju, K., Wang, Q., Yang, W., Gunter, C.A., Borisov, N.: Property inference attacks on fully connected neural networks using permutation invariant representations. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. pp. 619–633 (2018)
7. Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients-how easy is it to break privacy in federated learning? Advances in Neural Information Processing Systems **33**, 16937–16947 (2020)
8. Hitaj, B., Ateniese, G., Perez-Cruz, F.: Deep models under the gan: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. pp. 603–618 (2017)
9. Hu, H., Salcic, Z., Dobbie, G., Zhang, X.: Membership inference attacks on machine learning: A survey. arXiv preprint arXiv:2103.07853 (2021)
10. Jin, X., Chen, P.Y., Hsu, C.Y., Yu, C.M., Chen, T.: Cafe: Catastrophic data leakage in vertical federated learning. Advances in Neural Information Processing Systems **34**, 994–1006 (2021)
11. Kaissis, G.A., Makowski, M.R., Rückert, D., Braren, R.F.: Secure, privacy-preserving and federated machine learning in medical imaging. Nature Machine Intelligence **2**(6), 305–311 (2020)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Levy, M., Amit, G., Elovici, Y., Mirsky, Y.: The security of deep learning defences for medical imaging. arXiv preprint arXiv:2201.08661 (2022)
14. Mentzer, F., Toderici, G.D., Tschannen, M., Agustsson, E.: High-fidelity generative image compression. Advances in Neural Information Processing Systems **33**, 11913–11924 (2020)
15. Panchendrarajan, R., Bhoi, S.: Dataset reconstruction attack against language models. CEUR Workshop (2021)
16. Zhang, Y., Jia, R., Pei, H., Wang, W., Li, B., Song, D.: The secret revealer: Generative model-inversion attacks against deep neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 253–261 (2020)