Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning

Briland Hitaj*
Stevens Institute of Technology
bhitaj@stevens.edu

Giuseppe Ateniese
Stevens Institute of Technology
gatenies@stevens.edu

Fernando Perez-Cruz
Stevens Institute of Technology
fperezcr@stevens.edu

ABSTRACT

Deep Learning has recently become hugely popular in machine learning for its ability to solve end-to-end learning systems, in which the features and the classifiers are learned simultaneously, providing significant improvements in classification accuracy in the presence of highly-structured and large databases.

Its success is due to a combination of recent algorithmic breakthroughs, increasingly powerful computers, and access to significant amounts of data.

Researchers have also considered privacy implications of deep learning. Models are typically trained in a centralized manner with all the data being processed by the same training algorithm. If the data is a collection of users' private data, including habits, personal pictures, geographical positions, interests, and more, the centralized server will have access to sensitive information that could potentially be mishandled. To tackle this problem, collaborative deep learning models have recently been proposed where parties locally train their deep learning structures and only share a subset of the parameters in the attempt to keep their respective training sets private. Parameters can also be obfuscated via differential privacy (DP) to make information extraction even more challenging, as proposed by Shokri and Shmatikov at CCS'15.

Unfortunately, we show that any privacy-preserving collaborative deep learning is susceptible to a powerful attack that we devise in this paper. In particular, we show that a *distributed, federated, or decentralized deep learning* approach is fundamentally broken and does not protect the training sets of honest participants. The attack we developed exploits the real-time nature of the learning process that allows the adversary to train a Generative Adversarial Network (GAN) that generates prototypical samples of the targeted training set that was meant to be private (the samples generated by the GAN are intended to come from the same distribution as the training data). Interestingly, we show that record-level differential privacy applied to the shared parameters of the model, as suggested in previous work, is ineffective (i.e., record-level DP is not designed to address our attack).

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; **Distributed systems security**; **Software and application security**;

KEYWORDS

Collaborative learning; Security; Privacy; Deep learning

> It's not who has the best algorithm that wins.
> It's who has the most data.
>
> *Andrew Ng*
> *Self-taught Learning*

1 INTRODUCTION

Deep Learning is a new branch of machine learning that makes use of neural networks, a concept which dates back to 1943 [49], to find solutions for a variety of complex tasks. Neural networks were inspired by the way the human brain learns to show that distributed artificial neural networks could also learn nontrivial tasks, even though current architectures and learning procedures are far from brain-like behavior.

Algorithmic breakthroughs, the feasibility of collecting large amounts of data, and increasing computational power have contributed to the current popularity of neural networks, in particular with multiple (deep) hidden layers, that indeed have started to outperform previous state-of-the-art machine learning techniques [6, 29, 75]. Unlike conventional machine learning approaches, deep learning needs no feature engineering of inputs [45] since the model itself extracts relevant features on its own and defines which features are relevant for each problem [29, 45].

Deep learning models perform extremely well with correlated data, which contributed to substantial improvements in computer vision [47], image processing, video processing, face recognition [82], speech recognition [34], text-to-speech systems [64] and natural language processing [2, 15, 90]. Deep learning has also been used as a component in more complex systems that are able to play games [33, 42, 57, 60] or diagnose and classify diseases [16, 18, 26].

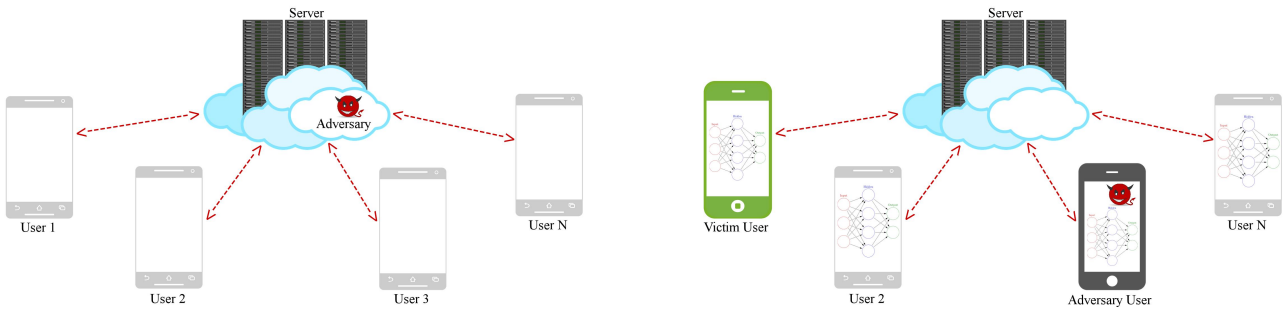
However, there are severe privacy implications associated with deep learning, as the trained model incorporates essential information about the training set. It is relatively straightforward to extract sensitive information from a model [4, 27, 28].

Consider the following cases depicted in Figure 1, in which N users store local datasets of private information on their respective devices and would like to cooperate to build a common discriminative machine. We could build a classifier by uploading all datasets into a single location (e.g., the cloud), as depicted in Figure 1 (a). A service operator trains the model on the combined datasets. This

*The author is also a PhD student at University of Rome - La Sapienza

CCS '17, October 30-November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-4946-8/17/10…$15.00
https://doi.org/10.1145/3133956.3134012



(a) Centralized Learning (b) Collaborative Learning

Figure 1: Two approaches for distributed deep learning. In (a), the red links show sharing of the data between the users and the server. Only the server can compromise the privacy of the data. In (b), the red links show sharing of the model parameters. In this case a malicious user employing a GAN can deceive any victim into releasing their private information.

centralized approach is very effective since the model has access to all the data, but it's not privacy-preserving since the operator has direct access to sensitive information. We could also adopt a collaborative learning algorithm, as illustrated in Figure 1 (b), where each participant trains a local model on his device and shares with the other users only a fraction of the parameters of the model. By collecting and exchanging these parameters, the service operator can create a trained model that is almost as accurate as a model built with a centralized approach. The decentralized approach is considered more privacy-friendly since datasets are not exposed directly. Also, it is shown experimentally to converge even in the case when only a small percentage of model parameters is shared and/or when parameters are truncated and/or obfuscated via differential privacy [77]. But it needs several training passes through the data with users updating the parameters at each epoch.

The Deep Learning community has recently proposed Generative Adversarial Networks (GANs) [30, 70, 72], which are still being intensively developed [3, 9, 32, 43, 56]. The goal of GANs is not to classify images into different categories, but to generate similar-looking samples to those in the training set (ideally with the same distribution). More importantly, GANs generate these samples without having access to the original samples. The GAN interacts only with the discriminative deep neural network to learn the distribution of the data.

In this paper, we devise a powerful attack against collaborative deep learning using GANs. The result of the attack is that any user acting as an insider can infer sensitive information from a victim's device. The attacker simply runs the collaborative learning algorithm and reconstructs sensitive information stored on the victim's device. The attacker is also able to influence the learning process and deceive the victim into releasing more detailed information. The attack works without compromising the service operator and even when model parameters are obfuscated via differential privacy. As depicted in Figure 1(a), the centralized server is the only player that compromises the privacy of the data. While in Figure 1(b), we show that any user can intentionally compromise any other user, making the distributed setting even more undesirable.

Our main contribution is to propose and implement a novel class of active inference attacks on deep neural networks in a collaborative setting. Our method is more effective than existing black-box or white-box information extraction mechanisms.

Namely, our contributions are:

(1) We devise a new attack on distributed deep learning based on GANs. GANs are typically used for implicit density estimation, and this, as far as we know, is the first application in which GANs are used maliciously.
(2) Our attack is more generic and effective than current information extraction mechanisms. In particular, our approach can be employed against convolutional neural networks (CNN) which are notoriously difficult for model inversion attacks [78].
(3) We introduce the notion of deception in collaborative learning, where the adversary deceives a victim into releasing more accurate information on sensitive data.
(4) The attack we devise is also effective when parameters are obfuscated via differential privacy. We emphasize that it is not an attack against differential privacy but only on its proposed use in collaborative deep learning. In practice, we show that differentially private training as applied in [77] and [1] (example/record-level differential privacy) is ineffective in a collaborative learning setting under our notion of privacy.

2 REMARKS

We devise a new attack that is more generic and effective than current information extraction mechanisms. It is based on Generative Adversarial Networks (GANs), which were proposed for implicit density estimation [30]. The GAN, as detailed in Section 5, generates samples that appear to come from the training set, by pitting a generative deep neural network against a discriminative deep neural network. The generative learning is successful whenever the discriminative model cannot determine whether samples come from the GAN or the training set. It is important to realize that both the discriminative and generative networks influence

each other, because the discriminative algorithm tries to separate GAN-generated samples from real samples while the GAN tries to generate more realistic looking samples (ideally coming from the same distribution of the original data). The GAN never sees the actual training set, it only relies on the information stored in the discriminative model. The process is similar to the facial composite imaging used by police to identify suspects, where a composite artist *generates* a sketch from an eyewitness *discriminative* description of the face of the suspect. While the composite artist (GAN) has never seen the actual face, the final image is based on the feedback from the eyewitness.

We use GANs in a new way, since they are used to extract information from honest victims in a collaborative deep learning framework. The GAN creates instances of a class that is supposed to be private. Our GAN-based method works only during the training phase in collaborative deep learning. Our attack is effective even against Convolutional Neural Networks which are notoriously difficult to invert [78], or when parameters are obfuscated via differential privacy with granularity set at the record level (as proposed in [77] and [1]). It works in a white-box access model where the attacker sees and uses internal parameters of the model. This in contrast to black-box access where the attacker sees only the output of the model for each particular input. It is not a limitation of our procedure because the purpose of collaborative learning is to share parameters, even if in a small percentage.

Once the distributed learning process ends, a participant can always apply a model inversion or similar attack to the trained model. This is not surprising. What we show in this paper is that a malicious participant can see how the model evolves and influence other honest participants and force them into releasing relevant information about their private datasets. This ability to deceive honest users is unique to our attack. Furthermore, truncating or obfuscating shared parameters will not help since our attack is effective as long as the accuracy of the local models is high enough.

We emphasize however that our attack does not violate differential privacy (DP), which was defined to protect databases. The issue is that, in collaborative deep learning, DP is being applied to the parameters of the model and with granularity set at the record/example level. However, the noise added to learning parameters will ultimately have to be contained once the model becomes accurate. Our attack works whenever the model can accurately classify a class and will generate representatives of that class. The way DP is applied in [77] and [1] can at best protect against the recovery of specific elements associated with a label that was indeed used during the learning phase. The results of our attack may or may not be regarded as privacy violations. Consider the following examples:

(1) The victim's device contains standard medical records. The GAN will generate elements that look like generic medical records, i.e., items from the same distribution of those in the training set. The attacker may learn nothing of interest in this case, and there is no privacy violation. However, if the victim's device contains records of patients with cancer then the attacker may see inexistent patients, but all with cancer. Depending on the context, this may be considered a privacy violation.
(2) The victim's device contains pornographic images. The GAN will generate similar scenes. While they may appear simulated, the information leaked to the adversary is significant. In other cases, our attack could be useful to law enforcement officials acting as adversaries. For instance, when the victim's device contains pedo-pornographic images or training material for terrorists.
(3) The victim's device contains speech recordings. The GAN will generate babbling, with lots of fictitious word-like sounds (comparable to WaveNet [64] when the network is trained without the text sequence), thus there is no privacy violation. However, it may be possible to infer the language used (e.g., English or Chinese) or whether the speaker is male or female, and this leaked information may constitute a privacy violation.
(4) The victim's device contains images of Alice. The GAN will generate faces that resemble Alice much like a composite artist generates a sketch of an eyewitness's memory of Alice. In our attack framework, the adversary will also collect all these drawings of Alice and falsely claim they are Eve's. This will force the local model within the victim's device to release more relevant and distinctive details about Alice's face, exacerbating the leakage. However, while many see this as a privacy violation, others may disagree since the adversary may not recover the exact face of Alice but only a reconstruction (see Figure 2) . On the other hand, if Alice wears glasses or has brown hair, then this information will be leaked and may constitute a privacy violation depending on the context. A further example is given in Figure 3, where DCGAN was run on the CIFAR-10 dataset [41] while targeting a class consisting of approximately 6,000 images containing various horses. Note that the class could be labeled 'jj3h221f' and make no obvious reference to horses. The images produced by the GAN will tell the adversary that class 'jj3h221f' does not contain cars or airplanes but animals (likely horses).

Differential privacy in collaborative learning is meant to protect the recovery of specific elements used during training. Namely, an adversary cannot tell whether a certain X was included in the training set (up to a certain threshold value). We circumvent this protection by generating an X' which is indistinguishable from X. In Figure 2, we show a real example of a face X along with X', the image generated by the GAN. Both images look similar even though X' is not X. While this does not violate DP, it clearly leads to severe privacy violations in many cases. Our point is that example/record-level DP is inadequate in this context, much like secure encryption against a chosen-plaintext attack (CPA) is inadequate in an active adversarial environment. There is nothing wrong with DP per se (as there is nothing wrong with CPA-secure encryption); clearly DP provides information-theoretic protection but it's important to set its level of granularity right. At record level, it is just not enough to protect sensitive information in collaborative learning against active adversaries. One can consider DP at different granularities (e.g., at user or device level) but this is not what is proposed in [77]. Researchers can keep arguing about the proper use of DP or what DP is supposed to protect [40, 53, 54, 58], but ultimately, in the context of this work, one should ask: Would I use a system that let

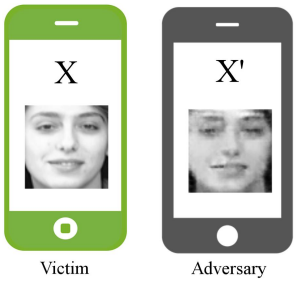


Figure 2: Picture of Alice on the victim's phone, X, and its GAN reconstruction, X'. Note that $X' \neq X$, and X' was not in the training set. But X' is essentially indistinguishable from X.

Figure 3: GAN-generated samples for the 'horse' class from the CIFAR-10 dataset

casual users recover images that are effectively indistinguishable from the ones in my picture folder?

The point is that collaborative learning for privacy is less desirable than the centralized learning approach it was supposed to improve upon: In centralized learning only the service provider can violate users' privacy, but in collaborative learning, any user may violate the privacy of other users in the system, without involving the service provider (see Figure 1).

3 IMPACT

Google adopts a centralized approach and collects usage information from Android devices into a centralized database and runs machine learning algorithms on it. Google has recently introduced *Federated Learning* [50, 51] to enable mobile devices to collaboratively learn a shared prediction model while keeping all the training data local. Devices download the current model from a Google server and improve it by learning from local data.

Federated learning appears to be the same as collaborative learning, and our attack should be equally effective. In the end, each device will download the trained model from the Google server, and the GAN will be able to operate successfully as long as the local model is learning.

In federated learning, it is possible to protect individual model updates. Rather than using differential privacy as in [77], Google proposes to use a secure aggregation protocol. The updates from individual users' devices are securely aggregated by leveraging secure multiparty computation (MPC) to compute weighted averages of model parameters [8] so that the Google server can decrypt the result only if several users have participated. We believe that this mechanism, as described in their paper, is ineffective against our attack architecture since we simply rely on the fact that local models have successfully learned. Their security model considers only the case in which Google is the adversary that scrutinizes individual updates. Therefore, they don't consider the point we raise in this paper that casual users can attack other users. This makes federated learning potentially even more dangerous than the centralized one it is supposed to replace, at least in its current form. Indeed, our assessment is based on the description given in an announcement and two research papers. We have had no access to the actual implementation of the system yet, and products tend to improve significantly over time.

Apple is said to apply differential privacy within a *crowdsourced learning* framework in future versions of iOS [35]. While we do not know the details, we hope our paper serves as a warning on the risks of applying differential privacy improperly in collaborative deep learning. Our adversary does not have to work for the service provider, but he is a regular user targeting another user, e.g., a celebrity or a politician.

4 RELATED WORK

Deep Learning has proven to be successful in various areas of computer science. The capability to learn, process and produce relevant information from large quantities of data, makes deep learning a good option for the cyber security domain as well. However, new and unique attacks have emerged that pose a serious threat to the privacy of the information being processed.

4.1 Attacks on Machine Learning Models

To the best of our knowledge, the first work that deals with extracting unexpected information from trained models is the one from Ateniese et al. [4] (released in 2011 and on arXiv in 2013 [4, 12]). There, the authors devised a meta-classifier that is trained to *hack* into other machine learning classifiers to infer sensitive information or patterns from the training set. For instance, they were able to extract ethnicity or gender information from trained voice recognition systems.

The work was later extended by Fredrikson et al. [27, 28] where they proposed model inversion attacks on machine learning algorithms by exploiting confidence information revealed by the model. For instance, when applied to facial recognition systems, they show that it is possible to reconstruct images about a particular label known to the adversary.

Recently, the work of Tramèr et al. [83] shows that *stealing* machine learning models is possible when taking into consideration only the predictions provided by the model. Membership inference attacks were developed by Shokri et al. [78]. Here, the adversary is given black-box access to the model and can infer whether a certain record was originally in the training set.

McPherson et al. [52] use deep learning to infer and reveal the identity of subjects behind blurred images. In their work, Papernot et al. [66] show that an adversarially crafted input can be fed to deep learning models and make them prone to error, i.e., make the model misclassify the input therefore producing incorrect outputs. For example, a STOP sign on the road can be subtly modified to look the same to human eyes, but that is classified as another sign by a trained model. The work was extended in [36, 44, 65, 87].

4.2 Privacy Preserving Machine Learning

Defense mechanisms against powerful adversaries were devised by Shokri and Shmatikov [77]. The authors introduce the concept of distributed deep learning as a way to protect the privacy of training data [85]. In this model, multiple entities collaboratively train a model by sharing gradients of their individual models with each other through a parameter server. Distributed learning is also considered in [17, 51, 59, 80, 89, 91]. Mohassel et al. [61] provide a solution for training neural networks while preserving the privacy of the participants. However, it deploys secure multiparty computation in the two-server model where clients outsource the computation to two untrusted but non-colluding servers. However, Shokri and Shmatikov [77] are the *first* to consider *privacy-preserving measures with the purpose of finding practical alternatives to costly multi-party computation (MPC) techniques.*

Google developed techniques to train models on smartphones directly without transferring sensitive data to the company's data centers [8, 51]. Microsoft developed CryptoNets [20] to perform deep learning on encrypted data and provide encrypted outputs to the users [86]. Ohrimenko et al. [63] developed data-oblivious machine learning algorithms trained on trusted processors. Differential privacy plays an important role in deep learning as shown in [1, 39, 77, 79].

4.3 Differential Privacy

Differential Privacy (DP) was introduced by Dwork [21]. Its aim is to provide provable privacy guarantees for database records without significant query accuracy loss. Differential privacy for big data was considered by Dwork et al. [23]. Several works have adopted DP as an efficient defense mechanism [5, 7, 11, 13, 19, 24, 25, 38, 55, 62, 67, 74, 88].

Collaborative deep learning proposed by Shokri and Shmatikov [77] uses DP to obfuscate shared parameters while Abadi et al. [1] propose to apply DP to the parameters during training. DP was used in deep auto-encoders in [69].

Covert channels, however, can be used to defeat DP-protected databases as shown in the work of Haeberlen et al. [37]. In general, privacy cannot be guaranteed if auxiliary information (outside the DP model) is accessible to the adversary [22]. At NDSS'16, it was shown by Liu et al. [48] that DP at a certain granularity is not effective in real-life scenarios where data such as social data, mobile data, or medical records have strong correlations with each other. Note that it's a matter of setting DP granularity right and DP is not being violated at all.

4.4 Privacy-Preserving Collaborative Deep Learning

A centralized approach to deep learning forces multiple participants to pool their datasets into a large central training set on which it is possible to train a model. This poses serious privacy threats, as pointed out by Shokri and Shmatikov [77], and distrustful participants may not be willing to collaborate.

Considering the security and privacy issues described above, Shokri and Shmatikov [77] introduce a new collaborative learning approach, which allows participants to train their models, without explicitly sharing their training data. They exploit the fact that optimization algorithms, such as Stochastic Gradient Descent (SGD), can be parallelized and executed asynchronously. Their approach includes a selective parameter sharing process combined with local parameter updates during SGD. The participants share only a fraction of their local model gradients through a Parameter Server (PS). Each participant takes turns and uploads and downloads a percentage of the most recent gradients to avoid getting stuck into local minima. This process only works if the participants agree in advance on a network architecture [77].

It is possible to blur the parameters shared with PS in various ways. Other than just uploading a small percentage of all the gradients, a participant can also select certain parameters that are above a threshold, within a certain range, or *noisy* in agreement with differential privacy procedures.

5 BACKGROUND

Supervised machine learning algorithms take labeled data and produce a classifier (or regressor) that it is able to accurately predict the label of new instances that has not seen before. Machine learning algorithms follow the inductive learning principle [84], in which they go from a set examples to a general rule that works for any data coming from the same distribution as the training set. Given independent and identically distributed (i.i.d.) samples from $p(\mathbf{x}, y)$, i.e., $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, 2, \ldots\}$, they solve the following optimization problem to find an accurate classifier:

$$\widehat{\theta} = \arg\min_{\theta \in \Theta} \sum_i L(f(\mathbf{x}_i; \theta), y_i) + \Omega(\theta), \tag{1}$$

where $\hat{y} = f(\mathbf{x}; \widehat{\theta})$ represents the learning machine, i.e., for any input $\mathbf{x}$ it provides an estimate for the class label y. $L(w, y)$ is a loss function that measures the error for misclassifying y by w. And $\Omega(\theta)$ is a regularizer (independent of the training data) that avoids overfitting. Supervised learning algorithms like Support Vector Machines (SVMs) [76], Random Forests [10], Gaussian Processes (GPs) [71] and, of course, deep neural networks [29] can be depicted by this general framework.

Deep neural networks are becoming the weapon of choice when solving machine-learning problems for large databases with high-dimensional strongly correlated inputs because they are able to provide significant accuracy gains. Their improvements are based on additionally learning the features that go into the classifier. Before deep learning, in problems that dealt with high-dimensional strongly correlated inputs (e.g., images or voice), humanly engineered features, which were built to reduce dimensionality and

correlation, were fed to a classifier of choice. The deep neural network revolution has shown that the features should not be humanly engineered but *learned from the data*, because the hand-coded features were missing out relevant information to produce optimal results for the available data. The deep neural network learns the useful features that make sense for each problem, instead of relying on best guesses. The deep neural network structures are designed to exploit the correlation in the input to learn the features that are ideal for optimal classification. The deep structure is needed to extract those features in several stages, moving from local features in the lower layers to global features at the higher layers, before providing an accurate prediction on the top layer. These results have become self-evident when datasets have grown in size and richness.

The learning machine $f(\mathbf{x};\theta)$ summarizes the training database in the estimated parameters $\widehat{\theta}$. From the learning machine and its estimated parameters, relevant features of the training database, if not complete training examples, can be recovered. So an adversary that wants to learn features from the original training data can do so if it has access to the learning machine. For example, SVMs store prototypical examples from each class in $\widehat{\theta}$ and GPs store all the training points, so there is no challenge there for an adversary to learn prototypical examples for each class in those classifiers. For deep neural networks, the relation between $\widehat{\theta}$ and the training points in $\mathcal{D}$ is more subtle, so researchers have tried to show that privacy is a possibility in these networks [77]. But the model inversion attack [27, 28] has proven that we can recover inputs (e.g., images) that look similar to those in the training set, leaking information to the adversary about how each class looks like. And as deep neural networks are trained with unprocessed inputs, these attacks recover prototypical examples of the original inputs.

It is important to emphasize that this is an intrinsic property of any machine-learning algorithm. If the algorithm has learned and it is providing accurate classification, then an adversary with access to the model can obtain information from the classes. If the adversary has access to the model, it can recover prototypical examples from each class. If sensitive or private information is needed for the classifier to perform optimally, the learning machine can potentially leak that information to the adversary. We cannot have it both ways, either the learning machine learns successfully, or data is kept private.

5.1 Limitations of the Model Inversion Attack

The model inversion attack works in a simple way [27, 28]: Once the network has been trained, we can follow the gradient used to adjust the weights of the network and obtain a reverse-engineered example for all represented classes in the network. For those classes that we did not have prior information, we would still be able to recover prototypical examples. This attack shows that any accurate deep learning machine, no matter how it has been trained, can leak information about the different classes that it can distinguish.

Moreover, the model inversion attack may recover only prototypical examples that have little resemblance to the actual data that defined that class. This is due to the rich structure of deep learning machines, in which broad areas of the input space are classified with high accuracy but something else is left out [31, 81]. If this is the case, the adversary might think he has recovered sensitive information for that class when he is just getting meaningless information. For example, we refer the reader to Figure 5 from [81], where six training images for a school bus, bird, a temple, soap dispenser, a mantis and a dog have been slightly tweaked to be classified as an ostrich (Struthio camelus), while they still look like the original image. In [31], the authors show in Figure 5 a procedure similar to the model inversion attack. A randomly generated image, plus gradient information from the deep belief network, produces a random looking image that is classified as an airplane. The structure of deep neural networks is so large and flexible that it can be fooled into giving an accurate label even though the image to a human looks nothing like it.

Thus any model inversion attack can obtain private information from a trained deep neural network, but it can land in an unrepresented part of the input space that looks nothing like the true inputs defined for each class. Extensive research in the ML community has shown that GAN generated samples are quite similar to the training data, thus the results coming from our attack reveal more sensitive information about the training data compared to the average samples or aggregated information one would expect from a model inversion type of attack.

5.2 Generative Adversarial Networks

One way to address the problem highlighted in [31, 81] is generating more training images so to cover a larger portion of the space. This can be accomplished through Generative Adversarial Networks (GANs) [30].

The GAN procedure pits a discriminative deep learning network against a generative deep learning network. In the original paper [30], the discriminative network is trained to distinguish between images from an original database and those generated by the GAN. The generative network is first initialized with random noise, and at each iteration, it is trained to mimic the images in the training set of the discriminative network. The optimization problem solved by the GAN procedure can be summarized as

$$\min_{\theta_G}\max_{\theta_D}\sum_{i=1}^{n_+}\log f(\mathbf{x}_i;\theta_D)+\sum_{j=1}^{n_-}\log(1-f(g(\mathbf{z}_j;\theta_G);\theta_D)) \quad (2)$$

where $\mathbf{x}_i$ are images from the original data and $\mathbf{z}_j$ are randomly generated images (e.g., each pixel distributed between 0 and 255 uniformly). Let $f(\mathbf{x};\theta_D)$ be a discriminative deep neural network that, given an image, produces a class label and let θ_D denote its parameters. Let $g(\mathbf{z};\theta_G)$ be a generative deep neural network, which given a random input produces an image.

The training procedure works as follows. First, we compute the gradient on θ_D to maximize the performance of the discriminative deep neural network. Hence $f(\mathbf{x};\theta_D)$ is able to distinguish between samples from the original data, i.e., $\mathbf{x}_i$, and samples generated from the generative structure, i.e., $\mathbf{x}_j^{\text{fake}} = g(\mathbf{z}_j;\theta_G)$. Second, we compute the gradients on θ_G, so the samples generated from $\mathbf{x}_j^{\text{fake}} = g(\mathbf{z}_j;\theta_G)$ look like a perfect replica of the original data[1].

[1]The generated data looks like the original data, but they are not copies of them.

The procedure ends when the discriminative network is unable to distinguish between samples from the original database and the samples generated by the generative network. The authors of the paper [30] prove the following theorem:

THEOREM 5.1. *The global minimum of the virtual training criterion in* (2) *is achieved if and only if* $p(\mathbf{x}) = p(g(\mathbf{z}; \theta_G))$.

The theorem shows that the adversarial game ends when the GAN is generating images that appear to come from the original dataset.

In [32], the author shows that in the infinite sample limit the generative network would draw samples from the original training distribution. But it also recognizes that the GAN procedure will not converge. In a recent paper [72], the authors have significantly improved the training of the GAN including new features to improve convergence to the density model.

6 THREAT MODEL

Our threat model follows [77], but relies on an active insider.

The adversary pretends to be an honest participant in the collaborative deep learning protocol but tries to extract information about a class of data he does not own. The adversary will also surreptitiously *influence* the learning process to deceive a victim into releasing further details about the targeted class. This *adversarial influence* is what makes our attack more effective than, for instance, just applying model inversion attacks [27] against the final trained model. Furthermore, our attack works for more general learning models (those for which a GAN can be implemented), including those on which model inversion attack is notoriously ineffective (e.g., convolutional neural networks).

Specifically, we consider the following scenario:

- The adversary works as an insider within the privacy-preserving collaborative deep learning protocol.
- The objective of the adversary is to infer meaningful information about a label that he does not own.
- The adversary does not compromise the central parameter server (PS) that collects and distributes parameters to the participants. That is, the parameter server, or the service provider in our example, is not under the control of the adversary. In our real-world example, the adversary is a full-fledged insider and does not have to work for the service provider.
- The adversary is *active* since he directly manipulates values and builds a GAN locally. At the same time, he follows the protocol specification as viewed by his victims. In particular, the adversary takes turns, follows the parameter selection procedures, uploads and downloads the correct amount of gradients as agreed in advance, and obfuscates the uploaded parameters as required by the collaborative learning process.
- As in [77], it is assumed that *all* participants agree in advance on a *common learning objective*. This implies that the adversary has knowledge of the model structure and, in particular, of the data labels of other participants.
- Unlike static adversaries as in model inversion [27], our adversary is allowed to be *adaptive* and work in real time while the learning is in progress. The adversary will be able to influence other participants by sharing specially-crafted gradients and trick participants into leaking more information on their local data. This is possible because the distributed learning procedure needs to run for several rounds before it is successful.

7 PROPOSED ATTACK

The adversary A participates in the collaborative deep learning protocol. All participants agree in advance on a common learning objective [77] which means that they agree on the type of neural network architecture and on the labels on which the training would take place.

Let V be another participant (the victim) that declares labels $[a, b]$. The adversary A declares labels $[b, c]$. Thus, while b is in common, A has no information about the class a. The goal of the adversary is to infer as much useful information as possible about elements in a.

Our insider employs a GAN to generate instances that look like the samples from class a of the victim. The insider injects these fake samples from a, as class c into the distributed learning procedure. In this way, the victim needs to work harder to distinguish between classes a and c and hence will reveal more information about class a than initially intended. Thus, the insider mimics samples from a and uses the victim to improve his knowledge about a class he ignored before training. GANs were initially devised for density estimation, so we could learn the distribution of the data from the output of a classifier without seeing the data directly. In this case, we use this property to deceive the victim into providing more information about a class that is unknown to the insider.

For simplicity, we consider first two players (the adversary and the victim) and then extend our attack strategy to account for multiple users. Each player can declare any number of labels, and there is no need for the classes to overlap.

(1) Assume two participants A and V. Establish and agree on the common learning structure and goal.
(2) V declares labels $[a, b]$ and A labels $[b, c]$.
(3) Run the collaborative deep learning protocol for several epochs and stop only when the model at the parameter server (PS) and both local models have reached an accuracy that is higher than a certain threshold.
(4) First, the Victim trains the network:
    (a) V downloads a percentage of parameters from PS and updates his local model.
    (b) V's local model is trained on $[a, b]$.
    (c) V uploads a selection of the parameters of his local model to PS.
(5) Second, the Adversary trains the network:
    (a) A downloads a percentage of parameters from the PS and update his local model.
    (b) *A trains his local generative adversarial network (unknown to the victim) to mimic class a from the victim.*
    (c) *A generates samples from the GAN and labels them as class c.*
    (d) A's local model is trained on $[b, c]$.

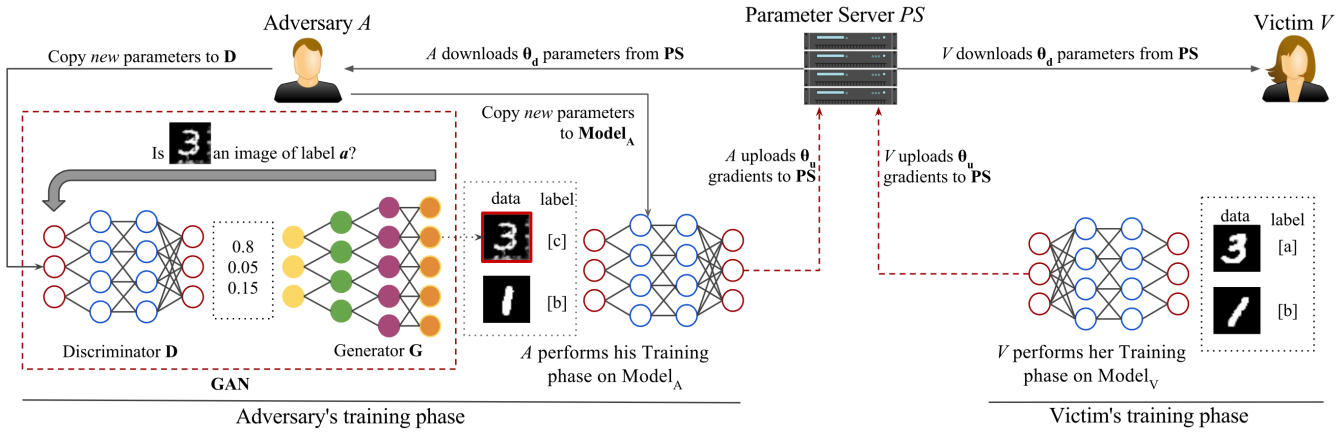


Figure 4: GAN Attack on collaborative deep learning. The victim on the right trains the model with images of 3s (class *a*) and images of 1s (class *b*). The adversary only has images of class *b* (1s) and uses its label *c* and a GAN to fool the victim into releasing information about class *a*. The attack can be easily generalized to several classes and users. The adversary does not even need to start with any true samples.

(e) *A* uploads a selection of the parameters of his local model to PS.

(6) Iterate between 4) and 5) until convergence.

The steps highlighted in 5b) and 5c) above represent the extra work the adversary perform to learn as much as possible elements of the targeted label *a*. The procedure is depicted in Figure 4. The generalization of the attack to multiple users is reported in Algorithm 1.

The GAN attack works as long as *A*'s local model improves its accuracy over time. Another important point is that the GAN attack works even when differential privacy or other obfuscation techniques are employed. It is not an attack on differential privacy but on its proposed use in collaborative deep learning. Though there might be a degradation in the quality of results obtained, our experiments show that as long as the model is learning, the GAN can improve and learn, too. Of course, there may always exist a setup where the attack may be thwarted. This may be achieved by setting stronger privacy guarantees, releasing fewer parameters, or establishing tighter thresholds. However, as also shown by the results in [77], such measures lead to models that are unable to learn or that perform worse than models trained on centralized data. In the end, the attack is effective even when differential privacy is deployed, because the success of the generative-discriminative synergistic learning relies only on the accuracy of the discriminative model and not on its actual gradient values.

8 EXPERIMENTAL SETUP

The authors of [77] provided us with their source code that implements a complete distributed collaborative learning system. Our attacks were run using their implementation of differential privacy.

8.1 Datasets

We conducted our experiments on two well-known datasets, namely MNIST [46] and AT&T dataset of faces [73] (a.k.a. Olivetti dataset of faces).

8.1.1 MNIST Dataset of Images. MNIST is the benchmark dataset of choice in several deep learning applications. It consists of handwritten grayscale images of digits ranging from 0 to 9. Each image is of 32 × 32 pixels and centered. The dataset consists of 60,000 training data records and 10,000 records serving as test data.

8.1.2 AT&T Dataset of Faces (Olivetti dataset). AT&T dataset, previously used also in the work of [27], consists of grayscale images of faces of several persons taken in different positions. The version used in our experiments consists of 400 images of 64×64 pixels.[2] The dataset contains images of 40 different persons, namely 10 images per person.

For these experiments, we did not conduct any pre-processing of the data. The only processing performed on the data was scaling every image to the [−1, +1] range, similar to [70]. This was done to adopt the state-of-the-art generator model of [70], which has a hyperbolic tangent *tanh* activation function in its last layer, thus outputting results in the [−1, +1] range as well.

8.2 Framework

We build our experiments on the Torch7 scientific computing framework.[3] Torch is one of the most widely used deep learning frameworks. It provides fast and efficient construction of deep learning models thanks to LuaJIT[4], a scripting language which is based on Lua[5].

8.3 System Architecture

We used a Convolutional Neural Network (CNN) based architecture during our experiments on MNIST and AT&T. The layers of the networks are sequentially attached to one another based on the

[2] http://www.cs.nyu.edu/roweis/data.html
[3] http://torch.ch/
[4] http://luajit.org
[5] https://www.lua.org

Algorithm 1 Collaborative Training under GAN attack

Pre-Training Phase: Participants agree in advance on the following, as pointed out also by [77]:

(1) common learning architecture, (model, labels etc.) {For ex. V declares labels $[a, b]$ and A labels $[b, c]$}
(2) learning rate, (lr)
(3) parameter upload fraction (percentage), (θ_u)
(4) parameter download fraction, (θ_d)
(5) threshold for gradient selection, (τ)
(6) bound of shared gradients, (γ)
(7) training procedure, (sequential, asynchronous)
(8) parameter upload criteria {cf. [77]}

Training Phase

for $epoch = 1$ to $nrEpochs$ **do**
  Enable user X for training
  User x downloads θ_d parameters from PS
  Replace respective local parameters on user x local model with newly downloaded ones
  if $(user\_type == ADVERSARY)$ **then**
    Create a replica of local $freshlyupdated$ model as D (discriminator)
    Run Generator G on D targeting class a (unknown to the adversary)
    Update G based on the answer from D
    Get n-samples of class a generated by G
    Assign label c (fake label) to generated samples of class a
    Merge the generated data with the local dataset of the adversary
  end if
  Run SGD on local dataset and update the local model
  Compute the gradient vector $(newParameters - oldParameters)$
  Upload θ_u parameters to PS
end for
return Collaboratively Trained Model {At the end of training, the adversary will have prototypical examples of members of class a known only to the victim}

$nn.Sequential()$ container so that layers are in a feed-forward fully connected manner.[6]

In the case of MNIST (Figure 15), the model consists of two convolution layers, $nn.SpatialConvolutionMM()$, where the $tanh$ function is applied to the output of each layer before it is forwarded to the max pooling layers, $nn.SpatialMaxPooling()$. The first convolutional layer has a convolution kernel of size 5×5 and it takes one input plane and it produces 32 output planes. Whereas the second convolutional layer takes 32 input planes and produces 64 output planes and it has a convolution kernel of size 5×5. After the last max pooling layer, the data gets reshaped on a tensor of size 256, on which a linear transformation is applied which takes as input the tensor of size 256 and outputs a tensor of size 200. Then a $tanh$ activation function is applied to the output, which is then followed by another linear transformation which takes as input the tensor of size 200 and outputs a tensor of size 11. We modify the output layer from 10 to 11, where the 11th output is where the adversary trains with the results generated by G. As in Goodfellow et. al [30], the 11th class is the class where the 'fake' images are placed. Further details are provided on Section 9. The last layer of the models is a LogSoftMax layer, $nn.LogSoftMax()$.

Images in the AT&T dataset of faces are larger (64×64). Therefore, we built a convolutional neural network (Figure 17) consisting of three convolution layers and three max pooling layers, followed by the fully connected layers in the end. As in the MNIST architecture, $tanh$ is used as an activation function. This model has an output layer of size 41, namely 40 for the real data of the persons and 1 as the class where the adversary puts the reconstructions for his class of interest. Since faces are harder to reconstruct than numbers, we implemented Algorithm 1 differently. For this case, the generator G queries the discriminator D more times per epoch (size of adversary's training data divided by batch size) to improve faster.

The $Generator(G)$ architecture used in MNIST-related experiments, Figure 16, consisted of 4 convolution layers corresponding to $nn.SpatialFullConvolution()$ from the torch 'nn' library. Batch normalization, $nn.SpatialBatchNormalization()$, is applied to the output of all layers except the last one. The activation function is the rectified linear unit function, $nn.ReLU()$. The last layer of the model is a hyperbolic tangent function, $tanh$, to set the output of G to the [-1, +1] range. Since AT&T images are larger (64x64), G has an additional (5th) convolution layer. The number of convolution layers needed were computed automatically using the techniques from [68]. G takes as input a 100-dimensional uniform distribution [14, 70], and converts it to a 32x32 image for MNIST or a 64x64 image for AT&T. As in [14], we initialized the weights of the generator with 0 mean and 0.02 standard deviation. While [70] applies this initialization function to both D and G, we do it *only* to G since D is the model that is shared among all participants.

Both architectures described above are represented in Figure 16 and 18 as printed out by Torch7.

We refer the reader to Appendix A for further details on the architectures provided by Torch7.

8.4 Hyperparameter Setup

For the MNIST-related experiments, we set the learning rate for both the collaboratively trained model and the discriminator model to $1e-3$, learning rate decay of $1e-7$, momentum 0 and batch size of 64.

For the AT&T-related experiments, we set the learning rate to 0.02 and a batch size of 32. Whereas, for the AT&T experiments concerning the multi-participant scenario, we used a batch size of 1. We kept the rest of the hyperparameters similar to the MNIST case. A learning rate of 0.02 worked better as it allowed more stochasticity in the process, thus allowing the model to converge faster.

The authors of DCGAN [70] use the *Adam* optimizer with a learning rate of 0.0002 and a momentum term β_1 of 0.5 as provided in the torch implementation of DCGAN [14]. We modified the process to use stochastic gradient descent (SGD) and, for this configuration, a learning rate of 0.02 for the generator worked better.

[6] https://github.com/torch/nn/blob/master/doc/containers.md#nn.Sequential

9 EXPERIMENTS

We now evaluate how well our GAN procedure can recover records from other participants. We focus our experiments on MNIST and AT&T datasets that contain images. In principle, however, our adversarial strategy can be extended to other types of data, such as music, audio, medical records, etc. We first compare our GAN attack against model inversion in a traditional setting. As mentioned before, model inversion has several limitations and may not be effective against certain types of neural networks. While this may be clear from a theoretical perspective, we also provide experimental evidence for this claim in the first experiment.

In the second set of experiments, we show how the GAN attack also works in the distributed setting in which the adversary is oblivious to the content of some, or all, labels, see Figure 7.

In the third set of experiments, we show that adding noise to the parameters of the deep neural network before they are uploaded to the parameter server does not protect against our GAN attack. In general, deploying record-level differential privacy to obfuscate the model parameters is ineffective against our attack. The efficacy of the GAN is only limited by the accuracy of the discriminator.

9.1 MI Attack vs. GAN Attack

In this first example, we compare the model inversion (MI) and the GAN attacks, and we provide them with all the data. The adversary has access to the fully trained models.

For the MI attack, we train a convolutional neural network on all 60,000 training examples of the MNIST dataset. We apply the model inversion attack in [27], once the deep neural network is trained. However, instead of approximating the derivatives as in [27], we collected the exact gradients computed by the model on the input given and the label (class) of interest. The results are shown in Figure 5. MI works well for MLP networks but clearly fails with CNNs. This is consistent with the work [78] where the authors attained similar results. It appears that MI is not effective when dealing with more complicated learning structures. While relevant information is in the network, the gradients might take us to an area of the input space that is not representative of the data that we are trying to recover.

For the GAN approach, we adopt the DCGAN architecture in [70], and its torch implementation from [14]. The model consists of the discriminator (D) in combination with the DCGAN generator (G). We made the generator model compatible with MNIST-type of images and used methods proposed in [68] so that our code could automatically calculate the number of convolution layers needed. We refer the reader to Section 8.3 for further details on the architectures. We ran the experiments 10-times (once per each class present in the MNIST dataset), and we let the models train until the accuracy reached by D was above 97%. We show the results in Figure 5.

Note a significant difference: In the GAN attack, the generative model is trained together with the discriminative model, while in MI, the discriminative model is only accessed at the end of the training phase. However, this type of *real-time access* to the model is what makes our attack applicable to collaborative deep learning.

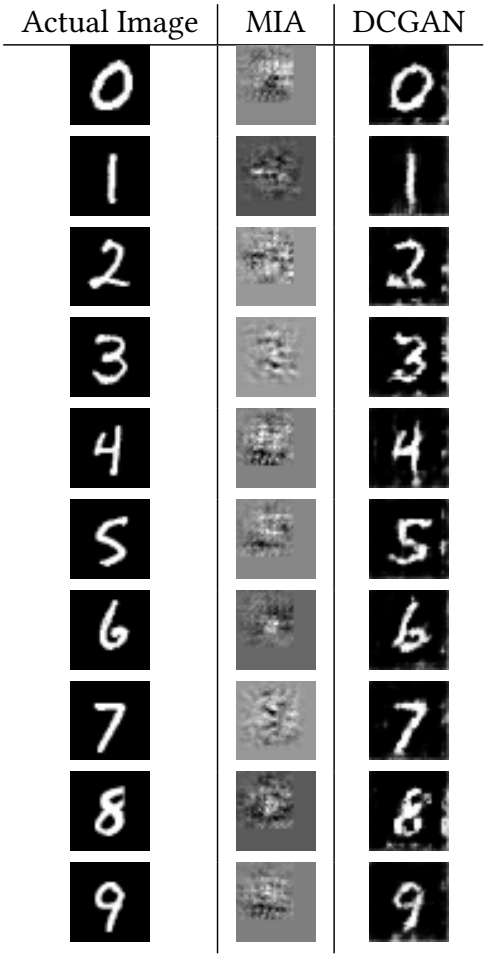


Figure 5: Results obtained when running model inversion attack (MIA) and a generative adversarial network (DCGAN) on CNN trained on the MNIST dataset. MIA fails to produce clear results, while DCGAN is successful.

9.2 GAN Attack on Collaborative Learning without Differential Privacy

Now we set the GAN attack in a collaborative environment like the one proposed in [77]. We use the model described in Section 7 and depicted in Figure 4.

9.2.1 Experiments on MNIST. Instead of using two labels per user, we use five labels for the first user and six labels for the second user. The first user has access to images of 0 to 4 (with label 1 to 5) and the second user, the adversary, has access to images of 5 to 9 (label 6 to 10). The adversary uses its sixth class to extract information on one of the labels of the first user.

The results are shown in Figure 6. For every retrieved image (bottom row), we placed above it an actual training image from the first user (we show the image that is closest in L1-norm). We have repeated the experiment with three different parameter settings. In (a), the users upload and download the entire model. In (b), the users

(a) $\theta_u = 1, \theta_d = 1$ (b) $\theta_u = 0.1, \theta_d = 1$ (c) $\theta_u = 0.1, \theta_d = 0.1$

Figure 6: Results for the GAN attack on a two-user scenario. Bottom row, samples generated by the GAN. Top row, samples from the training set closest to the ones generated by the GAN. (a) 100% parameters upload and download. (b) 100% download and 10% upload. (c) 10% upload and download.

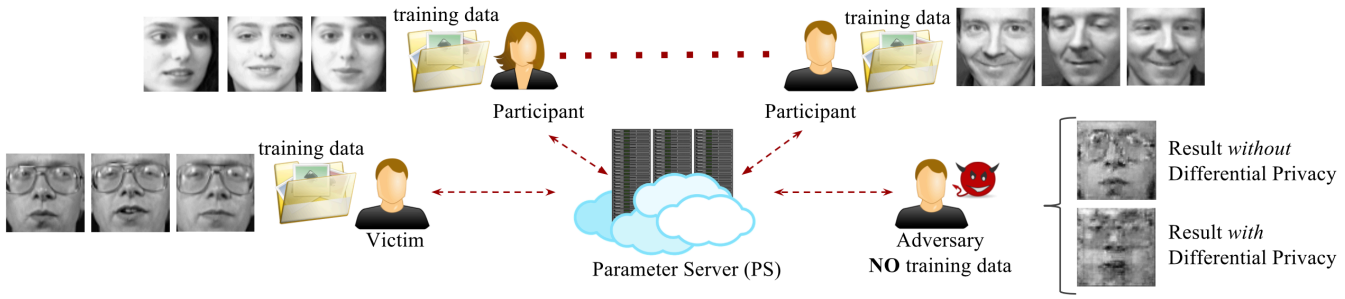


Figure 7: Collaborative deep learning with 41 participants. All 40 honest users train their respective models on distinct faces. The adversary has no local data. The GAN on the adversary's device is able to reconstruct the face stored on the victim's device (even when DP is enabled).

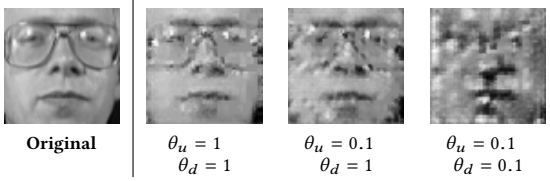


Figure 8: Experimental results on the AT&T Dataset with no DP. Unlike MNIST, images are noisier because this particular dataset is small and the accuracy of the model is significantly affected when upload rates are small.

download the full model, but only upload 10% of the parameters in each epoch. Finally, in (c), the upload and download is only 10%.

9.2.2 Experiments on AT&T. We performed similar experiments on the AT&T dataset which consists of faces from 40 different people. Initially, we tested the two-participant scenario, where one is the victim, and the other is the adversary. We assigned the first 20 classes to the first user and the remaining 20 classes to the adversary. An extra class is given to the adversary to influence the training process. We ran several configurations with different upload rates, see Figure 8. The results show the adversary can get considerably good reconstructions of the targeted face. Some images are noisier than others, but this can hardly be improved given that the accuracy of the model tends to stay low for this particular dataset.

We have also implemented a multi-participant scenario, see Figure 7, with 41 participants, 40 of which are honest and 1 is adversarial. Each honest participant possesses images pertaining to one class as training data, while the adversary has *no* training data of his own. Namely, the adversary only trains on the images produced by the generator (G). The results (with $\theta_u = 1, \theta_d = 1$) are very good even when differential privacy is enabled (Figure 7).

9.3 GAN Attack, No Influence vs. Influence on Collaborative Learning

One may wonder about the effect of the fake label to the collaborative learning. Recall that images generated by the generative model are placed into an artificial class to trick the victim into releasing finer details on the targeted class. We measured the effect of the adversarial influence, and we experimentally confirmed that its effect is remarkable: The learning gets faster, but also the information retrieved by the adversary is significantly better. We ran the experiments until the accuracy of the model on the testing set was above 97%, collaboratively training a CNN model. The datasets of both the adversary and the victim are separated from each other, and there are no labels in common.

In Figures 9 and 10, we show the result of the passive GAN attack with the standard GAN attack proposed in Section 7, when we are trying to recover, respectively, 0's and 3's from the first user. In the top row, we show the images from the passive attack with no influence and in the bottom row the images from the standard procedure with the influence of the artificial class. The effect of the

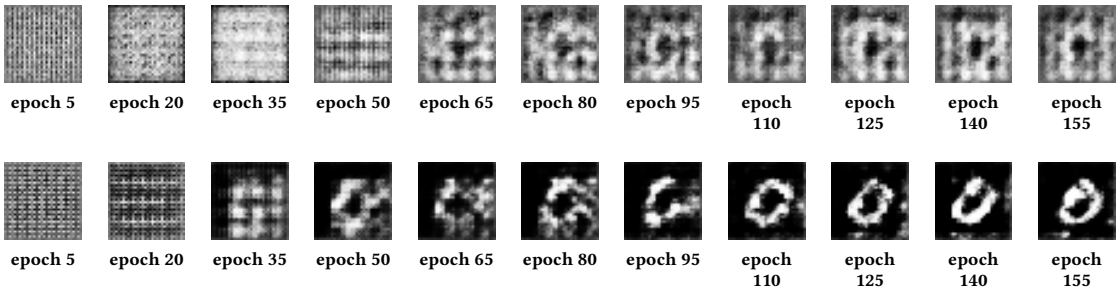


Figure 9: DCGAN with No influence vs. influence in Collaborative Learning for 0 (Zero)

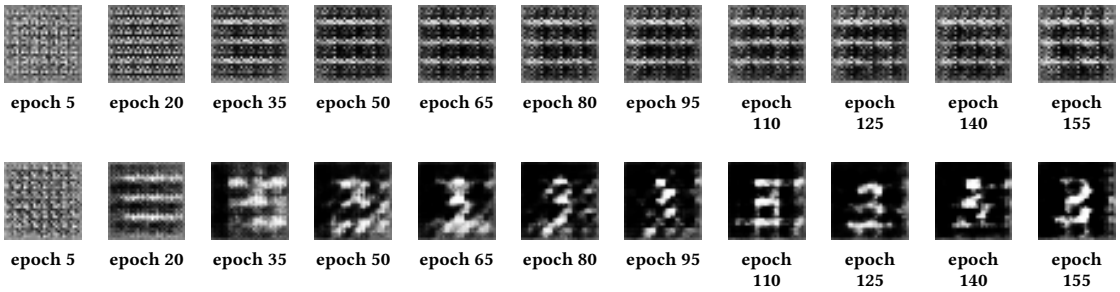


Figure 10: DCGAN with No influence vs. influence in Collaborative Learning for 3 (Three)

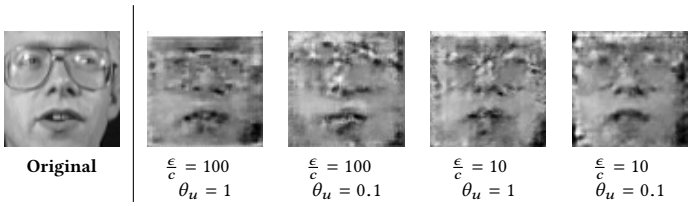


Figure 11: Experimental results on the AT&T Dataset with 100% download ($(\theta_d = 1)$ and DP enabled. Unlike MNIST, images are noisier because this particular dataset is small and the accuracy of the model is significantly affected when upload rates are small.

Figure 12: GAN Attack Results on the MNIST Dataset (left: original image, right: generated one) with DP Setup: $\frac{\epsilon}{c} = 0.01, \tau = 0.0001, \gamma = 0.001, \theta_u = 1, \theta_d = 1$. The value of ϵ is so small that the accuracy of the model does not increase. Since there is no learning, the GAN fails to produce clear results.

Figure 13: GAN Attack Results on the AT&T Dataset (left: original image, right: generated one) with DP Setup: $\frac{\epsilon}{c} = 0.01, \tau = 0.0001, \gamma = 0.001, \theta_u = 1, \theta_d = 1$. The value of ϵ is so small that the accuracy of the model does not increase. Since there is no learning, the GAN fails to produce clear results.

adversarial influence is evident, and images appear much clearer and crisper even after only 50 epochs per participant. During our experiments, we noticed that G starts producing good results as soon as the accuracy of the model reaches 80%.

(a) $\frac{\epsilon}{c} = 100$, $\theta_u = 1$, $\theta_d = 1$

(b) $\frac{\epsilon}{c} = 100$, $\theta_u = 0.1$, $\theta_d = 1$

(c) $\frac{\epsilon}{c} = 10$, $\theta_u = 1$, $\theta_d = 1$

(d) $\frac{\epsilon}{c} = 10$, $\theta_u = 0.1$, $\theta_d = 1$

Figure 14: Results for the GAN attack on a two-user scenario with Differential Privacy enabled. Bottom row, samples generated by the GAN. Top row, samples from the training set closest to the ones generated by the GAN.

9.4 GAN Attack on Differentially Private Collaborative Learning

It has been argued in [77] that differential privacy can be used to add noise to the parameters of the deep learning model *"to ensure that parameter updates do not leak too much information about any individual point in the training dataset."* (Quoted from [77].) The authors consider only a *passive* adversary and rely on differential privacy to mitigate possible leakages that might come from parameter updates. They highlight two cases of potential leakage: (i) the way how gradient selection is performed and (ii) actual values of the shared gradients. To address both of these issues, the approach in [77] relies on sparse vector technique [23]. For each epoch (iteration) of the collaborative learning process, they define a total privacy budget ϵ for each participant. This budget is split into c parts, where c is the total number of gradients that can be shared per epoch. A portion of gradients is randomly select such that they are above a threshold (τ). They dedicate $\frac{8}{9}$ of $\frac{\epsilon}{c}$ to the selection of the parameters and use the remaining $\frac{1}{9}$ to release the value. They rely on the Laplacian mechanism to add noise during selection as well as sharing of the parameters, in agreement with the allocated privacy budget.

To demonstrate that record-level differential privacy is ineffective against an active adversary, we ran the collaborative learning process between the two participants (A and V) with differential privacy enabled. We kept the datasets of the participants distinct: In MNIST experiments, V had only records of classes from 0 to 4 and A had records of classes from 5 to 9 plus the artificial class that A introduces. For the AT&T experiments, V has records for the first 20 classes in the dataset and A for the next 20 classes plus the artificial class as in Subsection 9.2. During our experiments we kept the download rate (θ_d) fixed at 100%, threshold (τ) at 0.0001 and the range (γ) at 0.001, similar to [77]. On Figures 11 and 14, we provide results for a privacy budget per parameter ($\frac{\epsilon}{c}$) of 100 and 10 and varying upload rate (θ_u). Even though it takes longer for the models to converge under the differential privacy constraints, our results demonstrate our claim, i.e., *as long as the training process is successful and the model is converging, G can generate good results.*

On the ϵ value. We observe that the ϵ in [77] is very large and the effect of differential privacy may be questionable. However, with small ϵ, the local models are unable to learn and collaborative learning fails completely. This is consistent with what is reported in [77]. Indeed, we ran our experiments with tighter privacy constraints. The generator failed to produce good results but because the local model were unable to learn at all. In Figure 12 and 13 we show an example where we set a tighter privacy bound, which translates into stronger differential privacy guarantees, and the GAN is ineffective. At the same time, this is expected since the local model and the one in the parameter server are unable to learn and collaborative learning is not happening. It is possible to use the techniques in [1] to bring ϵ down to a single-digit value. However, we stress again that our attack is independent of whatever record-level DP implementation is used. The GAN will generate good samples as long as the discriminator is learning (see Figure 2).

10 CONCLUSIONS

In this work, we propose and implement a novel class of active inference attacks on deep neural networks in a collaborative setting. Our approach relies on Generative Adversarial Networks (GANs) and is more effective and general than existing information extraction mechanisms. We believe our work will have a significant impact in the real world as major companies are considering distributed, federated, or decentralized deep learning approaches to protect the privacy of users.

The main point of our research is that collaborative learning is less desirable than the centralized learning approach it is supposed to replace. In collaborative learning, any user may violate the privacy of other users in the system without involving the service provider.

Finally, we were not able to devise effective countermeasures against our attack. Solutions may rely on secure multiparty computation or (fully) homomorphic encryption. However: (1) privacy-preserving collaborative learning was introduced as a way to avoid these costly cryptographic primitives [77], and (2) the solutions we explored based on them would still be susceptible to some forms

of our attack. Another approach is to consider differential privacy at different granularities. User or device-level DP would protect against the attacks devised in this paper. However, it's not clear yet how to build a real system for collaborative learning with device, class, or user-level DP (e.g., users behave and share data in unpredictable ways). Therefore, we leave this subject for future work.

ACKNOWLEDGMENT

We thank Martín Abadi, Matt Fredrikson, Thomas Ristenpart, Vitaly Shmatikov, and Adam Smith for their insightful comments that greatly improved our paper. We are grateful to the authors of [77] for providing us with the source code of their implementation of privacy-preserving collaborative deep learning.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 308–318.

[2] Ahmad Abdulkader, Aparna Lakshmiratan, and Joy Zhang. 2016. Introducing DeepText: Facebook's text understanding engine. (2016). https://tinyurl.com/jj359dv

[3] Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations (ICLR)*.

[4] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. 2015. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks* 10, 3 (2015), 137–150. https://arxiv.org/abs/1306.4447

[5] Gilles Barthe, Noémie Fong, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2016. Advanced probabilistic couplings for differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 55–67.

[6] Yoshua Bengio. 2009. Learning Deep Architectures for AI. *Found. Trends Mach. Learn.* 2, 1 (Jan. 2009), 1–127. https://doi.org/10.1561/2200000006

[7] Jeremiah Blocki, Anupam Datta, and Joseph Bonneau. 2016. Differentially Private Password Frequency Lists. In *NDSS'16*.

[8] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy Preserving Machine Learning. (2017).

[9] Diane Bouchacourt, Pawan K Mudigonda, and Sebastian Nowozin. 2016. DISCO Nets: DISsimilarity COefficients Networks. In *Advances in Neural Information Processing Systems*. 352–360.

[10] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[11] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*. Springer, 635–658.

[12] Jan Camenisch, Mark Manulis, Gene Tsudik, and Rebecca N. Wright. 2012. Privacy-Oriented Cryptography (Dagstuhl Seminar 12381). *Dagstuhl Reports* 2 (2012), 165–183. http://drops.dagstuhl.de/opus/volltexte/2013/3755/pdf/dagrep_v002_i009_p165_s12381.pdf

[13] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *Journal of machine learning research: JMLR* 12 (2011), 1069.

[14] Soumith Chintala. 2016. DCGAN.torch: Train your own image generator. (2016). https://github.com/soumith/dcgan.torch

[15] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12, Aug (2011), 2493–2537.

[16] Angel Alfonso Cruz-Roa, John Edison Arevalo Ovalle, Anant Madabhushi, and Fabio Augusto González Osorio. 2013. A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer Berlin Heidelberg, 403–410.

[17] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*. 1223–1231.

[18] DeepMind. 2016. DeepMind Health, Clinician-led. Patient-centred. (2016). https://deepmind.com/applied/deepmind-health/

[19] Ilias Diakonikolas, Moritz Hardt, and Ludwig Schmidt. 2015. Differentially private learning of structured discrete distributions. In *Advances in Neural Information Processing Systems*. 2566–2574.

[20] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Naehrig Michael, and John Wernsing. 2016. *CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy*. Technical Report MSR-TR-2016-3. http://research.microsoft.com/apps/pubs/default.aspx?id=260989

[21] Cynthia Dwork. 2006. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*. Springer Berlin Heidelberg, 1–12.

[22] Cynthia Dwork and Moni Naor. 2008. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality* 2, 1 (2008), 8.

[23] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[24] Cynthia Dwork and Guy N Rothblum. 2016. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887* (2016).

[25] Fabienne Eigner, Aniket Kate, Matteo Maffei, Francesca Pampaloni, and Ivan Pryvalov. 2014. Differentially private data aggregation with optimal utility. In *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, 316–325.

[26] Rasool Fakoor, Faisal Ladhak, Azade Nazi, and Manfred Huber. 2013. Using deep learning to enhance cancer diagnosis and classification. In *The 30th International Conference on Machine Learning (ICML 2013),WHEALTH workshop*.

[27] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1322–1333.

[28] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*. 17–32.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT Press.

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[31] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*. https://arxiv.org/pdf/1412.6572v3.pdf

[32] Ian J Goodfellow. 2014. On distinguishability criteria for estimating generative models. *arXiv preprint arXiv:1412.6515* (2014).

[33] Google DeepMind. 2016. AlphaGo, the first computer program to ever beat a professional player at the game of GO. (2016). https://deepmind.com/alpha-go

[34] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 6645–6649.

[35] Andy Greenberg. 2016. Apple's 'Differential Privacy' Is About Collecting Your Data—But Not Your Data. (2016). https://www.wired.com/2016/06/apples-differential-privacy-collecting-data/

[36] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2016. Adversarial Perturbations Against Deep Neural Networks for Malware Classification. *arXiv preprint arXiv:1606.04435* (2016).

[37] Andreas Haeberlen, Benjamin C. Pierce, and Arjun Narayan. 2011. Differential Privacy Under Fire. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*. USENIX Association, Berkeley, CA, USA, 33–33. http://dl.acm.org/citation.cfm?id=2028067.2028100

[38] Prateek Jain, Vivek Kulkarni, Abhradeep Thakurta, and Oliver Williams. 2015. To drop or not to drop: Robustness, consistency and differential privacy properties of dropout. *arXiv:1503.02031* (2015).

[39] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[40] Daniel Kifer and Ashwin Machanavajjhala. 2011. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 193–204.

[41] Nair Vinod Krizhevsky Alex and Hinton Geoffrey. [n. d.]. CIFAR-10 Dataset. ([n. d.]). https://www.cs.toronto.edu/~kriz/cifar.html

[42] Matthew Lai. 2015. Giraffe: Using deep reinforcement learning to play chess. *arXiv preprint arXiv:1509.01549* (2015).

[43] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*. 4601–4609.

[44] Pavel Laskov et al. 2014. Practical evasion of a learning-based classifier: A case study. In *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 197–211.

[45] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[46] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. 1998. The MNIST database of handwritten digits. (1998). http://yann.lecun.com/exdb/mnist/

[47] Yann LeCun, Koray Kavukcuoglu, Clément Farabet, et al. 2010. Convolutional networks and applications in vision.. In *ISCAS*. 253–256.

[48] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. 2016. Dependence Makes You Vulnerable: Differential Privacy Under Dependent Tuples. In *The Network and Distributed System Security Symposium 2016 (NDSS '16)*. 1322–1333. https://www.internetsociety.org/sites/default/files/blogs-media/dependence-makes-you-vulnerable-differential-privacy-under-dependent-tuples.pdf

[49] Warren S. McCulloch and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5, 4 (1943), 115–133. https://doi.org/10.1007/BF02478259

[50] Brendan McMahan and Daniel Ramage. 2017. Federated Learning: Collaborative Machine Learning without Centralized Training Data. (2017). https://research.googleblog.com/2017/04/federated-learning-collaborative.html

[51] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Ag?era y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *arXiv:1502.01710v5* (2016).

[52] Richard McPherson, Reza Shokri, and Vitaly Shmatikov. 2016. Defeating Image Obfuscation with Deep Learning. *arXiv:1609.00408* (2016).

[53] Frank McSherry. 2016. Differential Privacy and Correlated Data. (2016). https://github.com/frankmcsherry/blog/blob/master/posts/2016-08-29.md

[54] Frank McSherry. 2016. Lunchtime for Data Privacy. (2016). https://github.com/frankmcsherry/blog/blob/master/posts/2016-08-16.md

[55] Frank McSherry and Ilya Mironov. 2009. Differentially private recommender systems: building privacy into the Netflix Prize contenders. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 627–636.

[56] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. 2017. Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks. *arXiv preprint arXiv:1701.04722* (2017).

[57] Cade Metz. 2016. Google's GO victory is just a glimpse of how powerful ai will be. (2016). https://tinyurl.com/l6ddhg9

[58] Prateek Mittal. 2016. Differential Privacy is Vulnerable to Correlated Data Introducing Dependent Differential Privacy. (2016). https://tinyurl.com/l3lx7qh

[59] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. *arXiv:1602.01783* (2016).

[60] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv:1312.5602* (2013).

[61] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *IEEE Symposium on Security and Privacy*.

[62] Arjun Narayan, Ariel Feldman, Antonis Papadimitriou, and Andreas Haeberlen. 2015. Verifiable differential privacy. In *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 28.

[63] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. 2016. Oblivious Multi-Party Machine Learning on Trusted Processors. In *USENIX Security*.

[64] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A generative model for raw audio. *arXiv:1609.03499* (2016).

[65] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv preprint arXiv:1605.07277* (2016).

[66] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2015. The Limitations of Deep Learning in Adversarial Settings. *Proceedings of the 1st IEEE European Symposium on Security and Privacy* (2015).

[67] Manas Pathak, Shantanu Rane, and Bhiksha Raj. 2010. Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems*. 1876–1884.

[68] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M Álvarez. 2016. Invertible Conditional GANs for image editing. *arXiv preprint arXiv:1611.06355* (2016).

[69] NhatHai Phan, Yue Wang, Xintao Wu, and Dejing Dou. 2016. Differential Privacy Preservation for Deep Auto-Encoders: an Application of Human Behavior Prediction. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI*. 12–17.

[70] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *4th International Conference on Learning Representations*.

[71] C. E. Rasmussen and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.

[72] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2226–2234.

[73] Ferdinando S Samaria and Andy C Harter. 1994. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*. IEEE, 138–142.

[74] Anand D Sarwate and Kamalika Chaudhuri. 2013. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *IEEE signal processing magazine* 30, 5 (2013), 86–94.

[75] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.

[76] Bernhard Scholkopf and Alexander J Smola. 2001. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

[77] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. ACM, 1310–1321. https://doi.org/10.1145/2810103.2813687

[78] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P), Oakland*.

[79] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 245–248.

[80] Praveen Deepak Srinivasan, Rory Fearon, Cagdas Alcicek, Arun Sarath Nair, Samuel Blackwell, Vedavyas Panneershelvam, Alessandro De Maria, Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. 2016. Distributed training of reinforcement learning systems. (Feb. 4 2016). US Patent App. 15/016,173.

[81] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*. http://arxiv.org/abs/1312.6199

[82] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*. IEEE Computer Society, Washington, DC, USA, 1701–1708. https://doi.org/10.1109/CVPR.2014.220

[83] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security*.

[84] Vladimir Naumovich Vapnik and Vlamimir Vapnik. 1998. *Statistical learning theory*. Vol. 1. Wiley New York.

[85] Martin J Wainwright, Michael I Jordan, and John C Duchi. 2012. Privacy aware learning. In *Advances in Neural Information Processing Systems*. 1430–1438.

[86] Pengtao Xie, Misha Bilenko, Tom Finley, Ran Gilad-Bachrach, Kristin Lauter, and Michael Naehrig. 2014. Crypto-nets: Neural networks over encrypted data. *arXiv preprint arXiv:1412.6181* (2014).

[87] Weilin Xu, Yanjun Qi, and David Evans. 2016. Automatically evading classifiers. In *NDSS'16*.

[88] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. 2012. Functional mechanism: regression analysis under differential privacy. *Proceedings of the VLDB Endowment* 5, 11 (2012), 1364–1375.

[89] Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 116.

[90] Xiang Zhang and Yann André LeCun. 2016. Text Understanding from Scratch. *arXiv preprint arXiv:1502.01710v5* (2016).

[91] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. 2010. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*. 2595–2603.

A SYSTEM ARCHITECTURE

```
nn.Sequential {
  [input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) -> (10) -> (11) -> output]
  (1): nn.SpatialConvolutionMM(1 -> 32, 5x5)
  (2): nn.Tanh
  (3): nn.SpatialMaxPooling(3x3, 3,3)
  (4): nn.SpatialConvolutionMM(32 -> 64, 5x5)
  (5): nn.Tanh
  (6): nn.SpatialMaxPooling(2x2, 2,2)
  (7): nn.Reshape(256)
  (8): nn.Linear(256 -> 200)
  (9): nn.Tanh
  (10): nn.Linear(200 -> 11)
  (11): nn.LogSoftMax
}
```

Figure 15: Convolutional Neural Network Architecture used for MNIST related experiments, as printed by Torch. Note that the same architecture is used for both the collaboratively trained model and the local discriminator (D) model used by the Adversary

```
nn.Sequential {
  [input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) -> (10) -> (11) -> output]
  (1): nn.SpatialFullConvolution(100 -> 256, 4x4) without bias
  (2): nn.SpatialBatchNormalization
  (3): nn.ReLU
  (4): nn.SpatialFullConvolution(256 -> 128, 4x4, 2,2, 1,1) without bias
  (5): nn.SpatialBatchNormalization
  (6): nn.ReLU
  (7): nn.SpatialFullConvolution(128 -> 64, 4x4, 2,2, 1,1) without bias
  (8): nn.SpatialBatchNormalization
  (9): nn.ReLU
  (10): nn.SpatialFullConvolution(64 -> 1, 4x4, 2,2, 1,1) without bias
  (11): nn.Tanh
}
```

Figure 16: Generator Model Architecture used in MNIST experiments

```
nn.Sequential {
  [input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) -> (10) -> (11) -> (12) -> (13) -> (14) -> output]
  (1): nn.SpatialConvolutionMM(1 -> 32, 5x5)
  (2): nn.Tanh
  (3): nn.SpatialMaxPooling(3x3, 3,3)
  (4): nn.SpatialConvolutionMM(32 -> 64, 5x5)
  (5): nn.Tanh
  (6): nn.SpatialMaxPooling(2x2, 2,2)
  (7): nn.SpatialConvolutionMM(64 -> 128, 5x5)
  (8): nn.Tanh
  (9): nn.SpatialMaxPooling(2x2, 2,2)
  (10): nn.Reshape(512)
  (11): nn.Linear(512 -> 400)
  (12): nn.Tanh
  (13): nn.Linear(400 -> 41)
  (14): nn.LogSoftMax
}
```

Figure 17: Architecture of the Collaborative Model and the Discriminator (D) utilized in AT&T Dataset related experiments

```
nn.Sequential {
  [input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) -> (10) -> (11) -> (12) -> (13) -> (14) -> output]
  (1): nn.SpatialFullConvolution(100 -> 512, 4x4) without bias
  (2): nn.SpatialBatchNormalization
  (3): nn.ReLU
  (4): nn.SpatialFullConvolution(512 -> 256, 4x4, 2,2, 1,1) without bias
  (5): nn.SpatialBatchNormalization
  (6): nn.ReLU
  (7): nn.SpatialFullConvolution(256 -> 128, 4x4, 2,2, 1,1) without bias
  (8): nn.SpatialBatchNormalization
  (9): nn.ReLU
  (10): nn.SpatialFullConvolution(128 -> 64, 4x4, 2,2, 1,1) without bias
  (11): nn.SpatialBatchNormalization
  (12): nn.ReLU
  (13): nn.SpatialFullConvolution(64 -> 1, 4x4, 2,2, 1,1) without bias
  (14): nn.Tanh
}
```

Figure 18: Generator (G) Architecture used in AT&T Dataset related experiments, as printed by Torch7