Privacy Preserving Machine Learning-Based Methods for Synthetic Data Generation: A Survey and Review

Carlijn Nijhuis[a], Saba Amiri[a,*], Adam Belloum[a], Sander Klous[b], Leon Gommans[c]

[a]*Multiscale Networked Systems (MNS) Research Group, University of Amsterdam, 1098 XH Amsterdam, The Netherlands*
[b]*Complex Cyber Infrastructure (CCI) Research Group, University of Amsterdam, 1098 XH Amsterdam, The Netherlands*
[c]*Air France KLM*

Abstract

In recent years, it has been a given that machine learning systems need a huge amount of data to be trained and tuned. This is especially true for deep learning based methods and in use-cases such as computer vision. Gathering such a huge amount of data leads us to several challenges, two of the most important ones being lack of availability and privacy concerns. Synthetic data generation is one of the solutions to overcome the lack of availability. Although numerous data generation methods have already been reported in the literature with high levels of utility, the question of privacy still remains. Vulnerability of different machine learning models against attacks such as reconstruction, inversion and membership inference potentially puts the information used to train the data generation models at risk. Thus, methods of privacy preserving machine learning need to be utilized to ensure the privacy of the training data is preserved. This paper provides a thorough review of privacy preserving machine learning-based synthetic data generation methods. We first review the synthetic data generation methods and introduce privacy preserving methods employed to make machine learning systems private. Secondly, we provide an in-depth review of the literature on privacy preserving synthetic data generation methods and attempt to define the research space by putting these methods into different contexts for comparison and elaboration. We conclude by discussing open problems in the field and unexplored research areas.

Keywords: synthetic data generation, privacy preserving machine learning, differential privacy, generative models

1. Introduction

State-of-the-art machine learning systems have shown great potential and high degrees of success when applied to a variety of real world problems, including medical imaging [1, 2], speech recognition[3, 4] and autonomous driving[5, 6]. To train these systems effectively, in most cases large troves of information are needed. In specific use-cases, e.g. healthcare, recommender systems, personalization services, these systems would only be useful when access to massive amounts of potentially sensitive and private information is provided for training. This requirement poses several challenges in the areas of *privacy*, *accessibility* and *availability*.

The privacy challenge stems from the often realized need to access private and sensitive information to train machine learning systems. The first problem in this regard is related to laws and regulations protecting privacy of individuals in different domains. The General Data Protection Regulation (GDPR) of the European Union[7] and the Health Insurance Portability and Accountability Act in the United States[8] are examples of international legal regimes of this kind. More local privacy preservation measures such as ethics boards and local regulations add to the privacy guarantees of information, but severely limit the utility and usability of valuable information to train machine learning models. Methods such as anonymization try to solve this problem by removing "identifiers" such as name, address and social security number from data. But these methods have shown to not to be perfect, e.g. in [9] where Sweeney et al. successfully identified individuals in anonymized health record using publicly available information, including information of the governor of Massachusetts, William Weld. The problem gets more complicated when the data is consumed by a randomized mechanism such as a machine learning system. The main goal of training a machine model is to generalize over real world samples and it should ideally protect individuals' data samples. Generalization should smooth out discriminating details and prevent the model from overfitting, but in practice commonly used machine learning models tend to still overfit on some training samples, usually outliers, effectively memorizing them. Recent research on attacks against machine learning models have shown that information encoded in these models can be exploited to infer and extract sensitive information about the training dataset. Such attacks, as membership inference [10] and model inversion [11], successfully exploit the information encoded in the machine learning model to infer about the input data or recover sensitive features. Privacy preserving machine learning has gained a lot of momentum in recent years to explicitly deal with these types of privacy challenges. Mechanisms such as differential privacy[12] and cryptographic methods have been applied to different machine learning mechanisms in both centralized [13, 14, 15, 16, 17, 18]

*Corresponding author
Email address: s.amiri@uva.nl (Saba Amiri)

and federated[19, 20, 21, 22] architectures. While these methods have proven to provide varying levels of privacy, there usually is a trade-off involved in the form of loss of utility and computational and communication/time overhead.

The accessibility challenge refers to reachability of the training data by the machine learning model during its training phase and has become more prominent in recent years as with increases in capacity and availability of storage systems and also advances in distributed and cloud computing infrastructures, the data sources have become more distributed. The increase in resolution of the data as well as the number of samples being collected means that in many situations communication of the disparate datasets to a central location is not feasible. This is besides privacy rules and regulations mentioned before which effectively prohibit sharing of the data in many cases. Thus, training a machine learning model has transformed from a centralized paradigm to a decentralized one. The introduction of collaborative and federated learning [23] solves the problem of disparate data sources by moving the computations to local sites. In the simplest form of federated learning, in each round local models will be trained on the private dataset of each participant and the parameters will be send to a central orchestrator. The orchestrator will combine the parameters together and send back the updated parameter set to all participants. This training loop will be done for many iterations until the convergence criteria is met. This scheme solves the privacy problem of having to share private datasets in a central location for the centralized machine learning scheme. However, the risks mentioned above, e.g. privacy attacks against a trained machine learning model, are not mitigated. Furthermore, addition of a number of participants and a communication pipeline poses additional security and privacy challenges since each participant could potentially act as a malicious operator, trying to uncover the data from other participants or affect the outcome of the model by methods such as poisoning attacks[24].

The availability challenge is based on the simple fact that in many situations there might not be enough training data available. In many situations, we have too few samples to train a machine learning model effectively, forcing us to either underfit and lose utility or overfit, which leads to loss of generalizability. This problem could be extended to skewed datasets - where one or several label are scarce - and, in case of federated learning, non-i.i.d distributed data. There have been methods proposed to overcome these challenges such as methods that deal specifically with training machine learning models on small datasets [25, 26, 27] and few shot learning [28, 29, 30].

Synthetic data generation is one of the methods to mitigate some of the challenges mentioned above. First introduced by Rubin et al. [31], synthetic data generation proposes not releasing and/or utilizing actual data and using synthetic artificial data with the same statistical properties instead. The synthetic data fully expresses the underlying dataset used as a seed without disclosing actual information about the seed dataset. Synthetic data has been used widely in recent years to train state-of-the-art machine learning systems[32, 33, 34, 35].

Using synthetic data would resolve the problem of *availability* since we can have many more samples based on a few examples. Granted it does not completely mitigate the problem, since basically we cannot learn what we do not know. It at least has the potential of mitigating the problem of overfitting of a complex machine learning models' iterative learning process to some extent[36, 37]. It also solves the problem of *accessibility* since synthetic data could potentially be moved or even generated through an API access on another machine without having to actually communicate the original data. However the most important issue, namely *privacy* still remains. To ensure the preservation of the data privacy, we need to provide rigorous and provable theoretical and empirical guarantees.

In this paper, we review, evaluate and compare methods of *Privacy Preserving Synthetic Data Generation (PPSDG)*. Our work identifies and explains privacy preserving machine learning-based methods to generate synthetic data, compares them from different aspects and provides a holistic overview of the field both as a stand alone research subject and as part of the broader research effort to preserve the privacy of individuals while training state-of-the-art machine learning models on their private, personal data to provide crucial services to the masses. Our work could be used as a reference to motivate and guide researchers on different aspects of privacy preserving synthetic data generation and to promote future research in this area as well as encourage researchers in the broader domain of privacy preserving machine learning to consider privacy preserving synthetic data generation as a viable and crucial component of their privacy preserving frameworks.

The primary contributions of this paper can be summarized as follows:

- Providing a detailed overview of the machine learning-based synthetic data generation methods.
- Identifying and examining different aspects of these methods and provide projections of the reviewed methods against these perspectives, comparing and contrasting them.
- Identify and discuss different high level issues in our selected research area and provide a comprehensive overview on each topic.
- Uncovering research gaps and directions for future research on machine learning-based privacy preserving synthetic data generation methods and their use-cases.

The rest of this paper is organized as follows. Section 2 provides background information on data synthesis, basic machine learning methods used in reviewed synthetic data generation methods in this paper, the threats against these methods that justify the need for privacy preservation mechanisms, and basic definitions related to privacy in machine learning and privacy preservation methods. Section 3 defines a brief taxonomical classification of the reviewed methods to cement the big picture overview of the field in the mind of the reader. Next, section 4 provides an in-depth description of every machine learning-based synthetic data generation method available in the literature to the best of the authors' knowledge. Section 5 views the reviewed methods from different perspectives and provides an

in-depth comparison of the methods projected on these perspectives. Section 6 takes a step back and provides different discussion points from a high level point of view, analyzing different aspects of synthetic data generation methods as they fit into the big picture of the reasearch area. Finally we provide proposals for future research in Section 7 and conclude our work in Section 8.

2. Preliminaries

Before we dive into the details of privacy preserving data synthesis research in section 3, we will first discuss the concept of synthetic data, discussing in more depth the motivations for using synthetic data and giving an overview of the underlying machine learning mechanisms used to synthesize data, regardless of whether they are privacy preserving or not. Furthermore, some use cases of synthetic data are explored.

2.1. What Is Synthetic Data?

As the name suggests, synthetic data is artificially manufactured. It is not generated by documenting real-world events, but created by employing random processes which are controllable and based on the statistical characteristics of the original data [38]. More recently machine learning techniques have been leveraged for synthetic data generation. In this paper we focus on fully synthetic data, but for the sake of completeness we also mention partially synthetic data. By definition, *fully synthetic data* is data that does not contain any real data at all and is artificially built [38]. This allows for strong privacy protection, but the utility might be compromised. *Partially synthetic data* is a dataset that only replaces the sensitive features with synthetic ones, keeping the other parts of the original dataset as they were. Partially synthetic data has a higher disclosure risk than fully synthetic data because it still contains some actual data, but this is also why it might have higher utility depending on the use-case [38].

Fully synthetic data was first pitched by Rubin [31]. He noted that honoring confidentiality constraints when releasing micro-data (information at the level of individual entries) became more and more important, since with increasing demand for the release of micro-data for analysis, there was an increase of interest in the ethical and legal aspects of releasing privacy sensitive information. At the time, there were some efforts made on masking data to preserve the confidentiality of the data, but these methods relied on the users having knowledge about the used masking techniques and also having access to special statistical software [31]. Rubin came up with the idea not to release micro-data at all albeit masked, but only synthetic data, constructed using multiple imputation technique [31]. Simply put, the data holder randomly and independently samples data from the original set and then imputes the new dataset using models trained and fit on the sampled data and releases multiple versions of the synthetically generated dataset to the public.

2.2. Motivations for Using Synthetic Data

Synthetic data resolves the problems mention in Section 1, namely *privacy*, *accessibility* and *availability* to some extent.

In terms of *privacy*, it can help with preserving the privacy of the original dataset since the actual (possibly privacy sensitive) data is not used and the artificial dataset doesn't contain any of the original data [31]. In this context it is a form of data anonymization and a more potent alternative to data masking techniques [31]. The traditional anonymization techniques are not privacy preserving enough and are often vulnerable to reidentification techniques [39], the most infamous example of which being the Netflix competition, where researchers were able to obtain the identity of users in the released anonymized dataset using information found in the Internet Movie Database [40]. Synthetic data can be used to train a machine learning model in a private manner. It can be shared with other parties to collaborate or use third-party services like MLaaS to train models on the collective synthetic data without worrying about the data sharing and leakage. Synthetic data has been recognized as a method of achieving privacy by legal regimes such as the General Data Protection Regulation [41] in Europe [42].

Employing synthetic data generation in a machine learning pipeline also helps resolve some of the issues with *accessibility*. First, it could solve the problem if accessibility in a centralized scheme with distributed datasets, where you can't move the computation to the data owner site. This could be due to a number of reasons, among them lack of trust in running a training code provided by the central party and scarcity of compute resources. One other example use-case is autonomous driving, where it is difficult and dangerous to acquire data on some scenarios such as accidents and unwanted behaviour of the vehicle or the environment. These edge cases and rare occurrences can be generated synthetically to help improve the training model [43]. The synthetic data can be tailored to the needs that the situation demands, focusing more on specific types of scenarios [43].

Lastly, in situations where enough data simply is not available synthetic data can to some extent resolve the problem of *availability*. For example, when labelled data is scarce for a supervised machine learning approach, we can synthesize labels as well as other missing features and improve the dataset [44].

2.3. Machine Learning Building Blocks

The focus of this research is on privacy preserving machine learning-based synthetic data generation methods. The methods falling into this category to the best of our knowledge are all based on variants of deep learning. Common methods employed in deep learning-based synthetic data generation are the Generative Adversarial Networks (GAN) [45] and Variational Auto-Encoders (VAE) [46]. In general, all these methods use the original data as input to train upon and during the training process learn how to create synthetic data with the same statistical characteristics as the input data without including any of the original data and while maintaining privacy according to a specific definition - differential privacy is the de facto standard. These methods determine the density function of the attributes

in the data and try to estimate the parameters of these functions. Then the values of the synthetic data are randomly picked from these distributions. Instead of using the data for specific purposes such as classification, these methods learn the inherent patterns in the input dataset and try to mimic the statistical characteristics of the actual data. In this section we provide the building blocks of the reviewed machine learning-based synthetic data generation methods.

2.3.1. Deep Learning

Both the GAN and the VAE consist of two neural networks. To fully understand the training process of these two generative models, one also needs knowledge on the common training process of deep neural networks, like multi-layer perceptrons [47] or convolutional neural networks [48]. A Deep Neural Network (DNN) consists of an input layer, a certain number of hidden layers and an output layer [49]. These layers consist of neurons that are fully or partially connected to each other via an activation function. These connections have certain weights assigned that need to be tuned during training. Usually, a neuron calculates the sum of the values it receives from the neurons it is connected to in a previous layer. The activation function, which usually introduces non linearity to the network, determines whether the neuron is "activated" or not.

When training a DNN, the optimal values for the weights and the biases are approximated [49] usually through a backpropagation algorithm. The backpropagation algorithm consists of two passes: forward pass and backward pass. In the forward pass, the training data is passed through the network to predict an output, e.g. a label [49]. This output is then compared to the expected value of the output. Using a loss function, the divergence of the network's prediction from the expected output is calculated. This calculated loss is then used in the backward propagation pass [49]. The loss is passed backwards in the network, where each neuron receives a fraction of the loss relative to its contribution to the result. The weights and biases are then changed to minimize the loss using an optimization method. The most commonly used optimization method is usually a variant of the gradient descent method [49]. During gradient descent, gradients of the loss function with regards to weights are calculated and used to change the value of the weights in small increments [49]. The most common implementation of gradient descent in DNNs is Stochastic Gradient Descent (SGD). Because it is not usually computationally efficient to compute the gradients with regards to every single data item, stochastic gradient descent provides an estimation of the algorithm by adding randomness and adjusting the weights through the calculation of gradients for one data sample or a subset of input data (in case of mini-batch SGD).

2.3.2. Generative Adversarial Network

A popular technique for synthetic data generation using generative models is the generative adversarial network (GAN), proposed by Goodfellow et al. [45]. Since Goodfellow et al. proposed the concept of GAN, some improvements and variants have been proposed like the Wasserstein GAN (WGAN) by Arjosvky et al. [50], the WGAN with Gradient Penalty (WGAN-GP) [51], the Conditional GAN (CGAN) [52] and the Deep Convolutional GAN (DCGAN) [53]. The general mechanism is the same however among all these variants. There are two components to a GAN: a discriminator which is responsible for trying to determine whether a given data sample is real or generated; and a generator which is responsible for trying to generate data that is as realistic as possible [45]. These two components engage in a minimax game; the generator tries to generate data that the discriminator cannot distinguish from the real data (by trying to maximize the classification error of the discriminator) and the discriminator tries to improve itself in distinguishing the generated data from the real data (it tries to minimize the classification error). The formal definition of the game is [45]:

$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$.

The discriminator loss and generator loss resulting from the comparison by the discriminator is used to train the generator and the discriminator. In this way, the two components engaging in this minimax game help improve each other [45]. Eventually, the process is stopped when the real distribution and the estimated distribution have neared each other to such a degree that the samples taken from them have become more or less indistinguishable [45]. The process is shown in Figure 1.

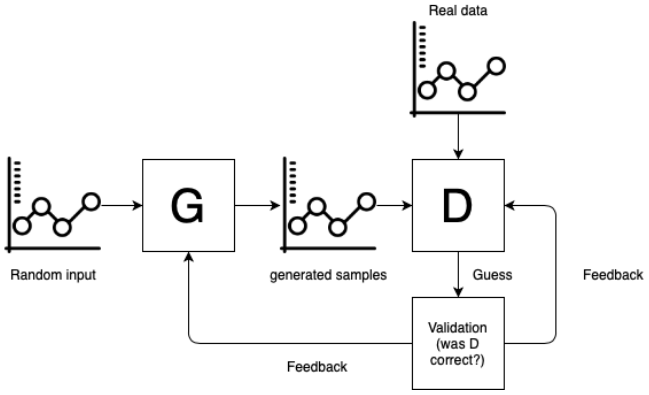

Figure 1: Workflow of a generic Generative Adversarial Network

2.3.3. Variational Auto-Encoder

The Variational Auto-Encoder (VAE) is a less popular, albeit still quite common method to generate synthetic data. It was proposed by Kingma et al. [46]. Similar to the GAN, the VAE also has two components. The difference is that these are not a generator and a discriminator but an encoder and a decoder [46]. The VAE is a variant of the normal Auto-Encoder (AE) [54].

An AE is used for dimensionality reduction of a dataset, which is the process of reducing the number of features to describe the data [55]. One would need less features in use-cases that benefit from low dimensional data, a simple example of which could be data storage. The encoder is responsible for mapping the high dimensional data to lower dimensions (in the so-called *latent space*) and then it is the decoder's task to reconstruct the high dimensional data from this encoded data [55].

This process of encoding and decoding can be lossy, meaning that some information can be lost in the process. The goal of the training process of an AE is to minimize this loss, but still maintain as low dimensional data as possible [55].

In theory, one could feed the decoder a value from the latent space and get a new data sample as a result. However, since no constraints are put on what types of characteristics the latent space should have, it could happen that the latent space is irregular [46]. The encoder and decoder are usually severely overfitted. This means that even though two values are close to each other in the latent space, they are decoded into two completely different values [46]. Normally, this would not be a problem if the AE is used for dimensionality reduction, but if you want to use the decoder to generate new samples by sampling random values from the latent space and decoding it, the resulting data could be unrepresentative, gibberish samples with low utility.

To mitigate this problem, the VAE was proposed [56]. It is almost the same as a regular auto-encoder, but it takes measures to regularize the latent space, which prevents overfitting [56]. It does this by encoding the data to a distribution over the latent space instead of encoding them to samples in the latent space. From this distribution, a point is sampled, which is then decoded. Unfortunately, this is not enough to ensure useful data generation. The encoder can still learn to map the data to distributions that are very far apart in the latent space or distributions with little to no variances. This way if you sample somewhere in between the distributions, the decoder will still have no idea what to do with it and will decode it to useless data. Thus, the mean of the distributions and the covariance matrix also need to be regularized [56]. This is often done by enforcing the condition that the distributions are close to a standard distribution. Therefore, the distributions could overlap a little, causing a gradient effect where the decoded results of a sample on the edge of two distributions being decoded to an average of the two values of the distributions [56]. Figure 2 depicts the basic mechanism of a VAE.

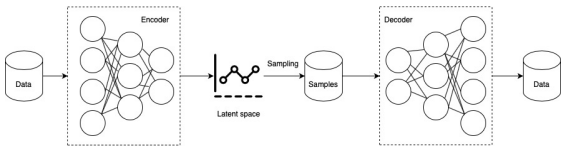

Figure 2: Variational Autoencoder encoding and decoding paths

2.3.4. Other Methods

Other than the methods already discussed, there are also other methods to generate synthetic data, such as traditional machine learning techniques, e.g. linear regression models, decision trees and random forest models [57]. The focus of this paper lies on the privacy preserving synthetic data generation models, the bulk of which employ deep learning-based generative models. Therefore, these techniques will not be explored. Nevertheless, the workflow of using synthetic data is the generally the same no matter which type of data generation you use. Real data is used to train a model (or determine a statistical distribution) that can generate synthetic data with the same statistical properties as the real data. After training, this model is used to generate new samples (or samples are randomly picked from the approximated distribution). This new dataset can then be used to train a different machine learning model for any number of tasks. An overview of the synthetic data generation and utilization workflow is depicted in Figure 3.

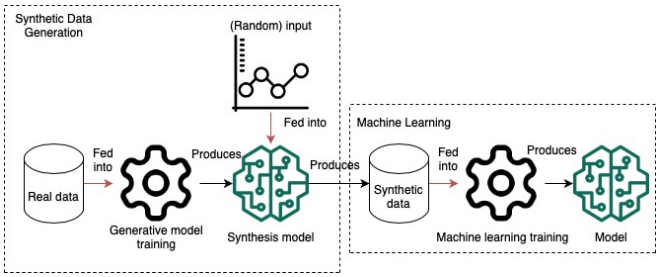

Figure 3: Generation and use of synthetic data in a typical machine learning setting

2.4. Threats Against Privacy in Machine Learning

Synthetic data in itself can be very useful for preserving the privacy of a dataset since no real data is included in the output. However, in this section we will show that the methods focusing merely on synthetic data generation are not enough to provide rigorous privacy guarantees. It is still possible in some cases to derive the original privacy sensitive data from the synthetically generated data. Therefore, we need to first, have a formalized and measurable definition of privacy and further, have mechanisms that satisfy the privacy levels required by different use-cases. In this section, we will introduce briefly the methods to exploit the privacy of machine learning models. Next we provide an overview of *differential privacy* as a widely used standard for machine learning privacy and briefly discuss methods usually employed to achieve differential privacy.

There are methods that exploit the inherent vulnerabilities of the underlying machine learning techniques used in synthetic data generation to infer about or reconstruct the training data. Most of these threats are based on the characteristics of the machine learning algorithms used, generally speaking the fact that they *learn* from the data. In specific cases models even memorize over certain data points. These threat have different goals, either trying to reconstruct the original dataset, reconstruct the trained model or infer about the original data, namely *reconstruction attacks*, *inversion attacks* and *inference attacks*. An overview of attacks can be seen in Figure 4.

2.4.1. Membership Inference Attacks

In membership inference attacks, the aim of the malicious party is to determine whether an individual sample was present in the original data used to train the model or not [58]. It does this by looking at the output of the machine learning model for a certain input and trying to find differences in the output of the model on input samples that were used in training and those that were not.

One example of such an attack is given by Shokri et al. [10]. Through the adversarial use of machine learning, they can determine whether a sample was part of the training data. They achieve this by training a inference model on the output of the target model that learns to distinguish the differences in the output of the target model for samples that were in the training data and for those that were not. They do this through black-box access - they have no inside knowledge about the specifics of the model and can only use its output. To train the inference model, they use a technique they call shadow training. In shadow training, a couple of shadow models are created which try to mimic the behavior of the victim's model. It should be noted that the access to these shadow models is white-box instead of black-box - meaning the attacker has inside knowledge about the model architecture and parameters. The attack model is then trained on these shadow models instead of on the victim's model directly. Through these techniques Shokri et al. show that they can successfully infer with and accuracy between 70% and 95% the membership of a sample was in the training data [10].

2.4.2. *Model Inversion Attacks*

Model inversion is another kind of attack on machine learning privacy that tries to infer the feature vectors of the model from responses provided by the model to adversarial queries (black-box) or from information contained in the model in the case where the model does not store the feature vectors explicitly (white-box) [58]. These types of attacks exploit the fact that machine learning models in most cases also return a confidence vector along with the result [58]. This confidence vector represents the probability that a result is correct, indicating how confident the model is about its answer.

An example of an implementation is the attack by Frederikson et al. [11]. They exploit the confidence vector, approaching the problem as an optimization problem, where the goal is to find an input that maximizes the confidence vector for a specific label.

2.4.3. *Reconstruction Attacks*

In reconstruction attacks, an adversary tries to reconstruct the data from the feature vectors of the machine learning model [58]. This attack is sometimes needed in combination with a model inference attack to determine the raw data from the derived feature vectors. The feature vectors are the representations of the data samples that can be fed to a machine learning model. This attack requires access to the feature vectors themselves. Some machine learning models store these feature vectors in their structure and are therefore accessible [58]. Generally speaking, because the data is not protected in the feature space, theoretically it can be reconstructed. A simple solution to this attack is to avoid using machine learning models that store the feature vectors in their parameters [58] but it still wouldn't completely mitigate the risk.

2.4.4. *Attacks on Collaborative/Federated Learning Systems*

Aside from the attacks that assume a centralized model access, e.g. through API access, there are also attacks exploiting the inner workings of the federated learning mechanism. One example of such attack is proposed by Hitaj et al. [59]. Instead of attacking the vulnerabilities in the machine learning model, this attack uses a vulnerability in the collaborative learning setting. Although only a small subset of the parameters is shared by each participant in the federated learning scheme, it is still possible to derive characteristics of a specific group of samples (belong to the same class, i.e. have the same label). Hitaj et al. do this by training a GAN to generate samples mimicking a certain class, but labeling it differently [59]. Therefore, a user that has data belonging to that class needs to provide more and more details of their class to distinguish it from the fake but similar class of the adversary. The more information on that class is shared by the victim, the better the GAN will be able to generate data belonging to that class. Eventually, if the GAN generates data that is almost indistinguishable from the original class, this can be used to generate samples on the class and determine some common statistical characteristics [59]. This is especially dire in use-cases such as face recognition, where each class represents a single person. In that case, the adversary can determine, through the images generated by the GAN, what the person looks like.

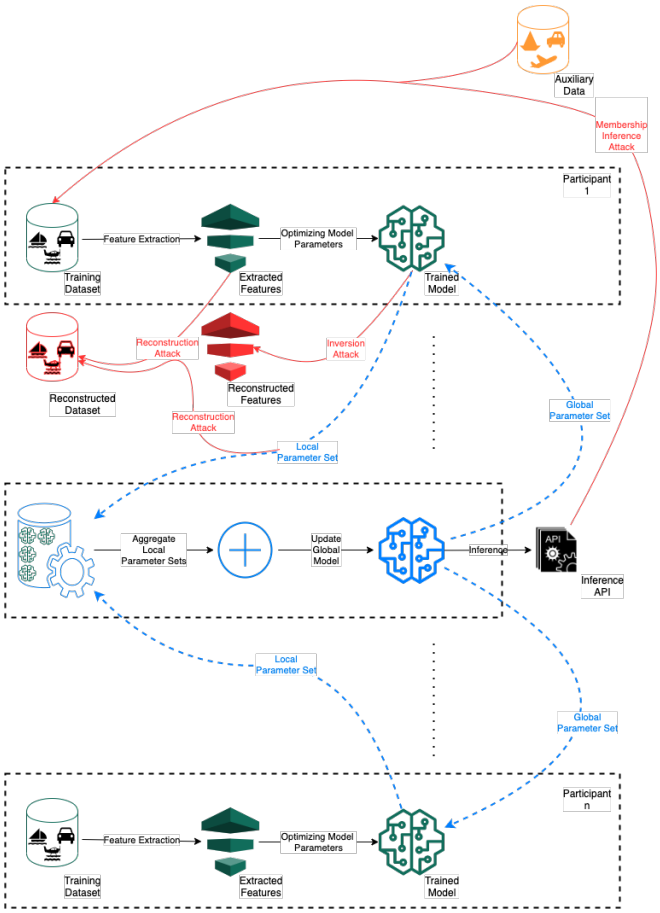

Figure 4: Attacks against machine learning systems, both local and federated. The boxes at the top and bottom depict two participants in a federated learning scheme. Adversarial attacks against privacy are depicted by red lines. The local learning process is lead by black lines, while the federated learning flow is shown using blue dashed lines.

Considering different attacks introduced in this section, we can conclude that the synthetic data generation model itself cannot be trusted to preserve privacy, because deep neural networks inherently contain information on the data used for training. We do not want to share or accidentally leak the model used to generate synthetic data, either white-box or black-box. Thus, the sharing of the generative models would not be possible at all. Furthermore, even if we do not share the model and only expose access to the model output through queries, by only sharing certain parameters in a collaborative setting, one can still determine some characteristics of the data. Therefore, we need ways to make the models inherently private which also allows for privacy in a collaborative settings. These solutions are not optional but necessary, otherwise synthetic data as a solution for preserving privacy will not be privacy preserving enough to serve its purpose.

2.5. Privacy Preserving Mechanisms for Machine Learning

To mitigate the threats introduced in Section 2.4, different solutions have been proposed in the literature, e.g. differential privacy [60], homomorphic encryption [61], federated learning [62][63], secure multi-party computation [64] and trusted execution environments [65]. The focus in this paper lies on the techniques that are most often used in the context of synthetic data generation, namely differential privacy and federated learning. Details and examples of implementations of these two paradigms in the synthetic data generation process are given in Sections 3 and 6. In the rest of this section, basics of differential privacy and federated learning are explained to provide the necessary preliminary knowledge to understand the privacy preserving data generation methods analyzed in this paper.

2.5.1. Federated Learning

Federated learning is a method that can help to solve part of the privacy preservation problem in a collaborative learning setting with users holding private datasets. It does so by negating the need to share and accumulate data in a central location and instead bringing the computation to the data [66]. In federated learning all participants can train a common machine learning model without having to exchange their private data. Usually, a central server orchestrates the entire process, while the training data remains decentralized. This server could prove to be a bottleneck however, or a point of potential privacy issues when the server is not fully trusted, more on that in section 3.1. The clients in this process first retrieve the model and training parameters from the server and partially train the model with their own data, for example using stochastic gradient descent. After training for a number of iterations, each node sends their local parameter set to the orchestrator. The orchestrator then computes an aggregate of the updates received from the clients and updates the model. This model is then sent to the clients again and the process continues until convergence. A global overview of a federated learning process as well as possible attacks against it have already been provided in Figure 4. The information exchanged with the central server is shaped in such a way as to contain as little information as possible needed for the machine learning task. Thus, the central server never sees the raw data.

The term federated learning was initially coined with mobile and edge devices in mind, where these devices are often present in huge numbers to collaborate on a common machine learning problem [66]. Nevertheless, interest in applying federated learning in a different setting, namely the setting where multiple organizations like hospitals work together each holding an accumulative private dataset has has been present from the start. The scenario of using mobile and edge devices is also called cross-device federated learning, while the scenario with multiple organizations is called cross-silo federated learning [66]. In this paper, the scenario of cross-silo federated learning is particularly interesting, where multiple different organizations, like hospitals, want to collaborate with each other to get a better generative model than if they had trained it on their own. Although of course, the setting where a lot of mobile and edge devices work together to achieve a common machine learning goal could also benefit from privacy enhancing technologies especially in the case when the data holders don't have the computational infrastructure to train different machine learning models on their local data, but training a privacy preserving synthetic data generation model *once* for repeated later usage is feasible.

As described before, most of the methods and definitions of federated learning assume that there is a central server that orchestrates everything. However recently, research efforts have been made to do federated learning in a decentralized fashion. With decentralized federated learning, there is no central server needed and the users coordinate the training process among themselves in a peer to peer fashion. An example of this is blockchain-based federated learning, where a blockchain is used to replace the central server to ensure the correct execution of the training algorithm in the case of malicious users [67].

From the privacy preservation point of view, federated learning in and of itself does not provide rigorous privacy guarantees and in its current form can still leak data through the exchanged model parameters, see Section 3.3. Therefore, it is necessary to combine it with differential privacy or other privacy enhancing technologies.

2.5.2. Differential Privacy

Differential privacy, the de facto privacy standard for machine learning in recent literature, provides formal privacy definitions that guarantee the privacy of individuals by providing them with plausible deniability. A simple example to show the main idea behind plausible deniability is the case of a researcher that interviews a group of people. Interviewees have to answer a simple yes or no question. Because the respondents might not want to give a truthful answer, the researcher asks the respondents to flip a coin. If the result is heads, they have to answer truthfully, otherwise they have to flip again and provide the result as their answer (yes becomes no, no becomes yes). This way the participants can deny inferences about their answer since the researcher does not know whether they had to lie or

not. This gives the participants the power of plausible deniability. In this example the flipping of the coins and coincidentally randomly flipping the answer of participants adds noise to the process. This is the main idea of differential privacy. The main idea of differential privacy is that for any two adjacent datasets differing in only one member, the adversary cannot distinguish between two random mechanisms applied on both by considering the probability distribution of their output [12]. This type of privacy is usually achieved through introduction of some kind of perturbation to the system, e.g. by addition of noise. Usually the more noise is added the higher the privacy is, but there will be a cost in terms of utility, making it into a tradeoff.

Definition of Differential Privacy. Formally, a randomized mechanism M is (ϵ, δ)-differentially private if for any output set S and any neighboring datasets D and $D^{'}$ differing in exactly one member [12]:

$$P(M(D) \in S) \leq e^{\epsilon} P(M(D^{'}) \in S) + \delta$$

.

The parameter ϵ, called the privacy budget, determines how strict the privacy is. The smaller the epsilon, the better the privacy. However smaller epsilon means lower utility for the mechanism. This can be seen by the fact that with perfect privacy ($\epsilon = 0$), the dataset would be completely randomized, i.e. the new data does not reveal any of the statistical characteristics of the original dataset. The parameter δ provides a relaxation of the original definition of ϵ-differential privacy to make it more flexible [68]. The addition of δ in the definition allows ϵ-differential privacy to fail for a small probability δ. The advice is to keep δ very small (smaller than the inverse of the database size $|D|$: $1/|D|$) [68]. This is to keep the risk of not protecting some individuals very small.

Another version of differential privacy has been proposed by Mironov et al., called Rényi Differential Privacy (RDP) [69]. RDP is a relaxation of ϵ-differential privacy based on the Rényi divergence [69]. It is comparable to (ϵ, δ)-differential privacy, but RDP is a stricter stronger privacy notion than (ϵ, δ)-differential privacy [69]. If a mechanism is RDP than it is also (ϵ, δ)-differentially private [69]. A mechanism M is RDP if the Rényi divergence of order a (Div_a) between the application of M on two adjacent datasets is no more than ϵ [69]:

$$Div_a(M(D)||M(D^{'})) \leq \epsilon$$

.

For two random variables X and Y that can take on n possible values with probabilities p and q, and order a, the Rényi divergence is defined as [70]:

$$Div_a(X||Y) = \frac{1}{a-1} log\left(\sum_{n}^{i=1} \frac{p_i^a}{q_i^{a-1}}\right)$$

There are also other definitions of differential privacy, e.g. Gaussian differential privacy [71] but since only the two regimes provided in this section have been utilized in privacy preserving synthetic data generation literature, we won't provide further information on these methods.

Common Noise Adding Mechanisms for Differential Privacy. There are two common mechanisms for adding noise: Gaussian and Laplacian [68]. As the name suggests, the Gaussian mechanism adds Gaussian noise to the algorithm, while the Laplacian mechanism adds Laplacian noise. The Laplacian mechanism preserves ϵ-differential privacy if the noise is added with scale $\Delta f/\epsilon$, where Δf is the sensitivity of the function it adds noise to. The sensitivity of a function is the maximum amount of effect addition or removal of one entry in a dataset can have on the output of the function. The Gaussian mechanism on the other hand preserves (ϵ, δ)-differential privacy if the scale of the noise satisfies the constraint

$$\sigma \geq \sqrt{2log(1.25/\delta)} \frac{\Delta_2 f}{\epsilon}$$

[12],

where $\Delta_2 f$ is a slightly different kind of sensitivity:

$$\Delta_2 f = \max_{D,D^{'}} ||f(D) - f(D^{'})||_2$$

.

Differential Privacy Budget Book Keeping. Considering the fact that many machine learning techniques use an iterative learning process in which data is accessed many times, keeping track of how much privacy budget has been spent by accessing the data is needed to calibrate the noise and to calculate the final privacy level of the system. To make sure that an adversary does not receive too much information from querying infinitely, one needs to keep track of a privacy budget. This privacy budget is the maximum privacy loss that is allowed. If the privacy budget is exceeded one can no longer query the dataset [12]. To keep track of the total privacy budget spent, so that epsilon is not exceeded, one needs privacy accounting. Moments Accountant [15] and Renyi's Differentially Private Accountant [69] are two accounting methods related to the provided definitions for differential privacy. These accountants keep track of the total accumulated privacy loss when repeatedly querying the data.

A big benefit of differential privacy is that post-processing does not affect the privacy guarantee. After making a process differentially private, no post-processing method is possible that can decrease the privacy [12]. This is called the post-processing theorem [12].

2.6. Use Cases

The two main motivations for synthetic data were already determined in section 2 to be data augmentation and privacy. To demonstrate the usefulness of synthetic data some example use cases are explored in this section.

There are a lot of areas where synthetic data could be useful. In health care for example, the synthetic data can help with both the issue of scarce data and with data that is privacy sensitive and cannot simple be shared or used in regular machine learning algorithms. A well known example of this is medGAN proposed by Choi et al. [72]. Choi et al. propose to use a combination of an autoencoder and a GAN to generate synthetic electronic health records.

Another area where synthetic data can help is with computer vision tasks, like surveillance systems or face recognition. In this area it is very difficult to collect sufficient and diverse data for training. Furthermore, some of the images or videos could be privacy sensitive as is the case with face recognition. An example of this is given by Asif et al. [73], where they show that with the use of only synthetic data it is possible to create a system for human fall detection. They first trained a generative model on real data, created synthetic data using this model and fed this data to another machine learning model to train it on to detect when a human has fallen (on camera).

Other than these two examples, synthetic data is also used in fraud detection systems, robotics and self-driving cars among others. So, one can see that synthetic data can be used in many different settings, for many different use-cases.

3. Privacy Preserving Data Generation

Now that the underlying machine learning methods and techniques, privacy definitions and threats and possible solutions for machine learning-based PPSDG have been discussed, this section will provide an explanation of the three main challenges that synthetic data faces regarding privacy.

In general, there are three main approaches when it comes to adding privacy to the generation process. One is to make the model used for the generation of synthetic data privacy preserving. Another is to use privacy preserving techniques to allow users in a distributed setting to safely collaborate. Thirdly, one needs to perturb the input before it is passed to the generative model in an MLaaS setting. It should be noted that one can also perturb the output of the entire process (the final model parameters). However, this last method results in low utility and does not guarantee privacy when the generative model itself is also fully known. So, this approach is not discussed further [15].

In this section, for the three main approaches discussed above, the techniques to make the process privacy preserving are briefly described to give a quick overview of the research area. This section is meant to give readers an initial overview of the challenges and methods in this reasearch area and act as a minimal framework to build knowledge upon it in the upcoming section.

3.1. Input Privacy

A scenario similar to the distributed setting, described in section 3.3, is the case of Machine Learning as a Service (MLaaS). With untrusted MLaaS providers, one cannot simply train the generative model on the real data, because this gives the MLaaS provider access to the privacy sensitive data as well. Similarly, in situations where one cannot assume a trusted third party to act as central server, clients are hesitant to share their data. A solution to this is to perturb the input data before passing it on to the generative model. Synthetic data itself is normally a solution to this problem, where one does not pass the real training data to a model stored at a provider, but synthetic data. However, when someone wants to generate the synthetic data at an MLaaS provider, this approach obviously does not work.

Research concerned with this scenario mostly focuses on perturbing the input in such a way that a provider or central server cannot derive anything useful from it. This can, for instance, be done by adding noise and new samples to the input.

Another possibility to hide the training data before passing it on to a generative model is homomorphic encryption. However to the best of our knowledge, these methods have not been extensively researched yet in the area of generative models and the solutions currently proposed are not applicable to arbitrary algorithms. Also due to the computationally-intensive nature of state-of-the-art cryptographic methods, no solutions have yet been provided with high computational efficiency. Therefore, we do not discuss it further, although it certainly is an important area for future research.

3.2. Model Privacy

One of the main focus points of research to make the process of synthetic data generation privacy preserving is trying to make the models used to generate the synthetic data private. Remember from Section 2.5.2 the post-processing principle stating that if a randomized method is made differentially private, further usage of it won't violate the privacy of the underlying data. In this way, an adversary is not able to derive the original data by looking at the released synthetic data and the generative model. If the generative model itself is privacy preserving one does not have to worry about keeping the generative model a secret, unless the model itself is considered and protected as intellectual property. The model no longer reveals enough information about the original data for an adversary to perform certain attacks like membership inference.

Some techniques have been proposed that can be integrated in a deep learning algorithm to make it privacy preserving. In short, they can be described as either trying to add noise to the gradients during the training process or as using ensembles to train (parts of) models on disjoint datasets. Abadi et al. proposed differentially private stochastic gradient descent (DPSGD) [15]. They noted that there were some existing techniques, specifically regularization techniques, that tried to prevent overfitting in deep learning models and coincidentally also helped with hiding details on the training set. However, the internal representations of neural networks are so large and complex that they still contain details on the training data. Therefore, Abadi et al. proposed a differentially private version of stochastic gradient descent, which offers protection against an adversary who has full knowledge of the training method and the values of the model parameters [15]. Their suggested approach consists of a DPSGD algorithm, a moments accountant and hyperparameter tuning. Their idea of a DPSGD algorithm does not entail adding noise only to the final parameters, since they say that adding over-conservative noise destroys the utility of the model. Therefore, they argue to add noise to the gradients throughout the learning process. In short, each iteration, they compute the gradient, clip it and then compute the average, add noise and then take a step in the opposite direction of this average noisy gradient. Finally, the privacy loss of the mechanism is calculated and kept track of by the moments accountant.

Another technique is Private Aggregation of Teacher Ensembles (PATE) proposed by Papernot et al. [74]. The idea behind PATE is to have an ensemble of teacher models, each

trained on a disjoint part of the private dataset. Then, this ensemble of teachers is used to train the student model in a semi-supervised way using a small set of public data. This student learns to mimic the ensemble by receiving the top vote from all their noisy votes for the public data. To keep track of the total amount of privacy loss, Papernot et al. [74] propose to use the Moments Accountant by Abadi et al. [15]. The privacy loss is determined by the number of queries the student model makes to the teachers ensemble. If the consensus in the teachers' vote is strong, then the privacy cost is small, since a change in one of the votes does not change the outcome of the consensus. In general, there is a trade-off in the number of teachers and the accuracy of the model. The more teachers there are, the stronger the consensuses are, but the more inaccurate the teacher models are, because they are each trained on less data. According to the authors, PATE has better accuracy on some datasets than DPSGD proposed by Abadi et al. [15], while guaranteeing more privacy. Another benefit is that the method is not dependent on the type of model used, but is described in a black-box fashion and can be used for all kinds of non-convex deep learning methods.

For both PATE and DPSGD, improvements have been proposed. For example the scalable PATE by Papernot et al. [75] that adjusts PATE to work well on a larger-scale learning task than it was originally tested on. However, the goal of this section is to provide a general overview of the types of solutions. An in-depth analysis of all the different implementations with possibly different versions of PATE and DPSGD will be performed in section 6. Therefore, these will not be further discussed here.

3.3. Collaborative Privacy

Another important scenario to take into account is when the data is distributed and this data cannot be shared among different participants, because of privacy reasons. In this case, the algorithm should be made differentially private in the distributed setting to allow different entities to safely collaborate without knowing the details of each other's data.

A solution to this is called Distributed Selective SGD (DSSGD), which was proposed by Shokri et al. [23]. It is also called Federated Stochastic Gradient Descent (FedSGD). The idea was that participants learn on their own data set and only share a small set of their model parameters with the others. The authors say that even with just a very small percentage of the parameters shared, one can get a significant increase in the model performance. This method was contradicted by Hitaj et al. however [59], showing that in a collaborative learning environment an adversary can influence the learning process and thereby obtain the private data from another user. They exploit the fact that DSSGD only protects the privacy of individual samples but not the privacy of groups. Therefore, if the adversary mimics one of the classes of samples from another user, but label it as another class, the other user will provide more and more details on the data in that class to differentiate it from the fake class created by the adversary. Thereby, providing the adversary with enough information to generate data belonging to that class even if differential privacy is used.

Therefore, most research is focused on trying to integrate federated learning into the generative algorithm in a privacy preserving way. In general, research in this area is focused on distributing the algorithm in such a way that no sensitive data is exchanged between the different users. They assume the existence of a central server that either delegates the work as is the case with Private FL-GAN by Xin et al. [76] or acts as an aggregator of the data in a sense, such as with AsynDGAN by Chang et al., where it acts as the central generator of the GAN model [77].

When there is no trustworthy central server available, decentralized federated learning could be a solution. The idea is that the participants coordinate the process themselves in a peer-to-peer fashion without the need of a central server to aggregate and orchestrate everything. However, the issue still persists that the other users cannot be trusted either. So, there still needs to be a solution to make the federated learning more private, before sharing information with the others. Especially maybe in a decentralized setting, where the information is directly shared with other potentially untrustworthy participants instead of with a central server. Now an adversary would directly receive information from the victims without the intervention of the server.

4. Review of Machine Learning-Based PPSDG Methods

In the previous section, section 3, a general introduction to the issues to be solved in privacy preserving data generation have been discussed. In this section, a brief explanation of each method to be analyzed is given. These methods are possible solutions to these issues. These explanations require knowledge on DPSGD [15] and PATE [74], which can be found in section 3. Furthermore, all other needed preliminary knowledge can be acquired in section 2.

A quick overview of the analyzed methods can be found in Table 1. In this table one can see the basic details, like the name of the method and the year it was published. Furthermore, the type of privacy technique it falls under is mentioned. Training perturbation means every method that adds noise somewhere during training and input perturbation means that the samples are perturbed before being fed to the training process. Methods that use PATE, Federated Learning or Kernel K-Means are marked as such. One can also see for what type of model the method was designed: GAN-based, AE-based or something else. Finally, the novelty of the method at the time of publication is given and a reference to the paper it belongs to is provided. An in-depth comparison is performed in section 5.

dPA was proposed by Phan et al. [78]. To our knowledge, it was one of the first methods to try and add differential privacy to the autoencoder. Similarly to DPSGD [15], it adds noise during the training process, specifically to the polynomial coefficients of the objective function. Instead of Gaussian noise, it adds Laplacian noise however. Furthermore, it does not use the Moments Accountant to keep track of the privacy budget, but a normalization layer on top of the hidden layer of the deep AE to ensure that the ϵ-differential privacy constraint is satisfied before moving on to the next hidden layer. To clarify, a

Method	Year	Privacy technique	Model Type	Novelty	Ref
dPA	2016	Training Perturbation	AE	Add DP to deep AE training	[78]
dp-GAN	2018	Training Perturbation	GAN	DP in GAN training with stability and scalability optimizations: parameter grouping, adaptive clipping, warm starting	[79]
DPGAN	2018	Training Perturbation	GAN	Add DP to GAN training with no public data needed	[80]
FP-GAN	2018	Training Perturbation	GAN	Add DP to GAN by adding noise in embedding space during forward pass in GAN training by using Gaussian noise layer in discriminator	[81]
Obfuscate	2018	Input Perturbation	Any	Include an obfuscation layer to transform input for MLaaS that preserves privacy of individuals and groups of samples	[82]
DPGM	2018	Training Perturbation + Kernel K- Means	Mixture	Using a mixture of generative models each trained on a different cluster + optimization for DPSGD by adaptively determining clipping threshold	[83]
DP-SYN	2018	Training Perturbation	AE	Use multiple DP AEs on subsets of data divided by label	[84]
PATE-GAN	2018	PATE	GAN	Use PATE in GAN without public data needed	[85]
DP-AuGM	2018	Training Perturbation	AE	Add DP to AE and generate data with the encoder	[86]
DP-VaeGM	2018	Training Perturbation	AE	Use multiple DP VAEs on subsets of data divided by label	[86]
GANobfuscator	2019	Training Perturbation	GAN	Optimization for DPSGD that improves stability and scalability by monitoring change of gradient magnitudes and dynamically adjusting the pruning bounds	[87]
EDP-GAN	2019	Training Perturbation	GAN	Add DP to GAN by adding noise to the output of the discriminator	[88]
DP-CGAN	2019	Training Perturbation	GAN	Optimization for DPSGD by clipping gradients of real and fake data separately + also generates labels + uses RDP accountant	[44]
cdp-GAN	2019	Training Perturbation	GAN	Can create continuous, categorical and time-series data + uses optimization clipping decay	[89]
FL-GAN	2020	Training Perturbation + Federated Learning	GAN	Add DP to federated learning with GANs	[76]
AsynDGAN	2020	Federated Learning	GAN	Propose asynchronous collaborative learning solution	[77]
PriVAE	2020	Training Perturbation	AE	Optimization for DP VAE by using term-wise DPSGD	[90]

Table 1: Global overview of PPSDG approaches

deep AE is an AE with more than one layer. One can see it as stacking multiple autoencoders on top of each other. dPA was specifically designed with human behavior prediction tasks in mind.

dp-GAN was proposed by Zhang et al. [79] as one of the first methods that tried to integrate (ϵ, δ)-differential privacy with GANs, specifically for data synthesis tasks. It implements DPSGD [15] by adding Gaussian noise to the gradients calculated in the backward pass when training the discriminator and using the Moments Accountant to keep track of the privacy loss. This is sufficient to enforce differential privacy, because of the post-processing theorem. If the weights of the discriminator are differentially private then the output from which the generator learns is also differentially private and therefore the generator is differentially private. Zhang et al. [79] also implemented some optimization strategies to improve the training scalability and stability. One of them is parameter grouping that tries to optimize the clipping and perturbation part of the training. Normally, one would group the gradients of all the parameters together to compute the norm for clipping. Parameter grouping does this in a more clever way. Zhang et al. experimented with two types of grouping: weight-bias separation and weight clustering. Weight-bias separation is the idea to differentiate between the bias and weight parameters, where the bias gradients are then grouped together for clipping. Weight clustering creates clusters of the weight gradients. Adaptive clipping is another optimization and it needs public data to work. The idea with adaptive clipping is to constantly monitor the magnitude of the gradients. The clipping bounds are then determined based on the average magnitudes. A batch is randomly sampled from the public data and the clipping bounds of each parameter are set to be the average gradient norm with respect to that

batch. This adaptive clipping leads to faster convergence of the training process and higher utility. Finally, warm starting is used to improve the convergence rate, which also needs public data. Zhang et al. [79] first train the model without privacy constraints on the public data to initialize it and then with privacy constraints on the private data.

DPGAN was proposed by Xie et al. [80]. They add Gaussian noise to the gradients calculated in the backward pass when training the discriminator to enforce (ϵ, δ)-differential privacy which is sufficient due to the post-processing theorem. Specifically, they try to add differential privacy to the Wasserstein GAN (WGAN) [50], but Xie et al. say that their method can be used in other GAN frameworks as well. They use the idea of DPSGD to implement privacy and use the Moments Accountant to keep track of the total privacy loss. Specifically, their idea is to add noise to the gradients of the Wasserstein distance with respect to the training data instead of the usual Jensen-Shannon divergence that is used to determine the loss in vanilla GAN.

FP-GAN was proposed by Triastcyn et al. [81]. Instead of adding Gaussian noise to the gradients of the discriminator loss during training like with DPSGD, they add noise to the embedding space during forward pass by adding a Gaussian noise layer in the discriminator in the GAN. This noise layer causes the output of the discriminator and therefore also the weights of the discriminator to be differentially private. It makes the weights differentially private, because the weights are updated using the gradients calculated in the backward pass using the calculated loss. This loss is calculated using the discriminator output and hence if this output is differentially private, all other calculations and computations on this output are also differentially private, because of the post-processing theorem. The method uses the Moments Accountant [15] to keep track of the privacy budget and the privacy loss.

Obfuscate was proposed by Zhang et al. [82]. Zhang et al add privacy by adding noise to the input of the model. They propose to add an obfuscation layer to the entire training workflow that can preserve both the privacy of individual samples and that of entire groups of similar samples. The authors mention that most approaches only focus on preserving the privacy of individual samples, but not on the privacy of entire groups. Therefore, their Obfuscate method preserves privacy in two ways: adding Gaussian noise to the individual samples and adding more samples to a group. The latter makes sure that the statistical properties of groups of samples are protected. Specifically, they do this by randomly picking a sample from a group, converting the feature values into its negative and adding some noise. By converting the feature values to the negative, the statistical properties are effectively averaged out.

DPGM was proposed by Acs et al. [83]. It adds a private Kernel K-Means algorithm to the generation process along with DPSGD [83]. They use a combination of k generative neural networks that each train on their local partition of the data using an improvement of DPSGD. This data is divided using a differentially private kernel k-means algorithm. The privacy budget is tracked with the Moments Accountant [15]. Their reason for training the data divided over multiple different generative neural networks is that multiple networks trained on the data can generate better samples than one model trained on the entire dataset since they were trained on similar data (due to kernel k-means). It prevents the combination of different data clusters in a generated sample. An improvement on DPSGD is that they determine the clipping threshold adaptively each iteration based on the data. Each iteration of SGD, one model is chosen uniformly at random, depending on how big their assigned cluster is, to run one iteration of DPSGD on a sample of a certain batch size from its own subset of the data. The synthetic dataset is generated by mimicking the training process. One of the created models is selected randomly using a probability that reflects how often the model was chosen during training. This model is then used to generate one synthetic sample, after which the process is repeated until the required number of samples are obtained.

DP-SYN was proposed by Abay et al. [84]. It first divides the data by label, meaning that if there are k different labels, there are k different partitions. Each such partition is then used to train an AE using DPSGD with some optimizations regarding gradient calculation (computed for each training instant instead of for batches) and gradient clipping bounds that reduces the sensitivity of the gradients and stabilizes the training process. The encoded samples are then used in a differentially private version of the iterative expectation maximization algorithm DP-EM [91] to detect different latent patterns in these samples and to generate new samples with similar patterns (from the same latent space). These new samples are then decoded, creating new data, where these decoded samples are added to the synthetic dataset. Finally, they use the Moments Accountant to track the privacy loss [15].

PATE-GAN was proposed by Jordon et al. [85]. PATE-GAN focuses on exploiting PATE [74] for privacy preserving data synthesis [85]. Some modifications of the PATE framework were needed to be able to use it for GANs. Specifically, they use PATE in the discriminator. The discriminator is trained with a PATE mechanism, meaning that there are a number of teacher discriminators and one student discriminator. The teacher discriminators are trained on their loss with respect to the generator, the generator with respect to the student discriminator and the student discriminator with respect to the teacher discriminators. The outputs of the teachers are combined in a so-called teacher vote aggregation, where these votes are added and one single vote is the result. During this aggregation, noise is added to make the final result of the teacher vote aggregation less dependent on individual votes. This final vote is then shared with the student discriminator. The main adjustment that Jordon et al. make is that where PATE requires public unlabeled data, their method does not, because as they say, access to public data is not always achievable. Therefore, the student discriminator is not trained on public unlabeled data, but only on the outputs from the generator.

DP-AuGM was proposed by Chen et al. [86]. It uses one (ϵ, δ)-differentially private AE and needs public data to be able to generate data. They make the AE private by implementing DPSGD [15], which means adding Gaussian noise to the calculated gradients during SGD to make the weights that are adjusted with these gradients less dependent on the individual data

samples, and using the Moments Accountant [15] to keep track of the privacy loss. DP-AuGM does not need the public data for training, but for generating the data, because Chen et al. chose to use the encoder as generator [86]. Therefore, the encoder needs real data to encode and this creates the synthetic data.

DP-VaeGM was proposed by Chen et al. [86]. Chen et al. use multiple VAE models, where each VAE is responsible for generating the data of one specific class [86] in the data. The data is divided by label and each group is then used to train one VAE in a (ϵ, δ)-differentially private way, like DPSGD [15], where noise is added to the gradients during the backward pass phase to make the model differentially private. The privacy loss is tracked with the Moments Accountant [15]. Then to generate data, samples from the Gaussian distribution are fed into the models and these generated data samples are then joined together by taking the union of all the separate generated datasets from each VAE.

GANobfuscator was proposed by Xu et al. [87]. It uses a technique similar to DPSGD in the discriminator and the Moments Accountant [15] on the Wasserstein GAN [50] along with a gradient pruning strategy to improve the stability of the training process. The main idea of GANobfuscator is to use a combination of noise and gradient pruning to maintain both training stability and data quality, while preserving privacy. To ensure training stability and to increase the quality of the generated images, Xu et al. proposed an optimization scheme that uses adaptive pruning to monitor changes in the gradient magnitudes and to adjust the pruning bounds. The pruning bound is a hyperparameter in the algorithm of GANobfuscator that when too small causes excessive truncation of the gradients and when too large causes overestimating the sensitivity. This pruning bound is determined using a random sampled batch of public data. The bound is calculated as the average gradient norm with respect to that batch.

EDP-GAN was proposed by Triastcyn et al. [88]. EDP-GAN adds Gaussian noise to the output of the discriminator of the GAN by employing the Gaussian noise mechanism in the second-to-last layer in the discriminator. There, the input of the layer is clipped and perturbed by the noise. In other words, the activation function of the layer is adjusted to clip and add noise. This way, the generator receives a noisy output to train on. It does not use the Moments Accountant or any privacy accounting during training, but an ex-post analysis built on the ideas of empirical differential privacy [92] and on-average KL privacy [93], using KL divergence estimation and the Chebyshev's inequality to find the bound on the privacy loss.

DP-CGAN was proposed by Torkzadehmahan et al. [44]. It uses a different relaxation of ϵ-differential privacy than (ϵ, δ)-differential privacy, namely, Rényi differential privacy. Therefore, instead of the Moments Accountant, DP-CGAN uses the Rényi Differential Privacy Accountant, which provides a tighter bound on the privacy budget, meaning that less noise is needed to achieve a certain level of privacy. They enforce differential privacy by adding Gaussian noise to the gradients of the discriminator of the conditional GAN during the backward pass in training. As an improvement, Torkzadehmahan et al. clip the gradients of discriminator loss on real and fake data separately instead of on the sum of both. According to the authors, this allows for a better control of the sensitivity of the model to the real data, which also increases utility. Finally, DP-CGAN also generates the labels as well as data and the synthetic data can therefore be used for supervised learning tasks without the need for manual labeling.

cdp-GAN was proposed by Frigerio et al. [89] as an expansion of dp-GAN. It allows for the generation of time series, continuous and discrete data. Frigerio et al. do this by using Long Short Term Memories as the generator for stream data and a multilayer perceptron for discrete data. Furthermore, their clipping decay method does not need public data. The idea of clipping decay is to reduce the clipping parameter over time to account for the fact that gradients decrease over time as well and thus to avoid the gradients being hidden by the noise. Otherwise, it is very similar to dp-GAN, using some form of DPSGD and Moments Accountant to preserve (ϵ, δ)-differential privacy.

FL-GAN was proposed by Xin et al. [76]. It combines federated learning with (ϵ, δ)-differential privacy adding noise similar to DPSGD [15]. They add Gaussian noise to the gradients of the discriminator during the training process and use the Moments Accountant to keep track of the privacy loss. Each participant in the process has its own dataset. In the beginning, one participant receives an initialized model from a central server and trains its own dataset on this model using DPSGD [15]. The discriminator and the generator are privacy preserving and hence do not leak any information to the other users and the central server. This model is returned to the server. The server then moves on to another client and passes on the model from the previous client to the other. This is done until all clients have trained their data on the model. The authors chose serial training because it results in better models and less data access, according to them.

AsynDGAN was proposed by Chang et al. [77]. It uses a couple of distributed discriminators and one central generator. The discriminators are located where the data is and they interact with the central generator that tries to learn to generate data that resembles all the distributed data. The discriminators do not communicate among themselves and the only thing that is communicated between the generator and a discriminator are auxiliary data, synthetic data and discriminator loss. The central generator generates data that is passed on to the discriminators. These discriminators try to determine whether it is real or not, which is validated and this feedback is sent back through the network. In other words, the loss is calculated and the discriminator is updated with regards to this loss. The central generator receives from each discriminator the discriminator loss and will be updated with regards to this loss as well.

PriVAE was proposed by Takahashi et al. [90]. It is one of the few methods that focuses on VAEs. Takahashi et al. use a new version of DPSGD, because according to them vanilla DPSGD [15] is not suitable for VAEs. Vanilla DPSGD increases the sensitivity, which causes a large amount of noise to be needed that in turn decreases utility. The reason that this happens is because DPSGD assumes that the micro-batch size is 1. To solve this, Takahashi et al. propose term-wise DPSGD that creates random gradients based on the compositions of the loss

terms. Specifically, term-wise DPSGD decomposes the term of the loss function into two groups and composes a noisy gradient that guarantees differential privacy per group. For each group, the gradients are computed in a stochastic way, the gradients are clipped and Gaussian noise is added. The noisy gradients are computed separately for sample-wise terms and batch-wise terms. In the final phase, the noisy gradients are combined and the parameters of the model are updated. Term-wise DPSGD does not only work for VAEs, but can also be used in other deep learning models.

5. Comparison of PPSDG Approaches

Now that a general overview of the research area and methods has been given, these methods are compared in this section. A quick overview of all analyzed methods can be found in Table 1. This table shows which privacy technique was used, in which year it was published, what type of generative model the approach works on and what the main innovation was of the approach compared to the others at the time of publication.

Some initial observations by looking at Table 1 will be explored in section 5.1. Then, we'll compare the methods looking at the different ways to generate data (using GANs, AEs or ensembles) in section 5.2. Thirdly, the methods are compared looking at how and if they provide differential privacy in section 5.3. Fourthly, some methods rely on public data, these will be analyzed in section 5.4. Finally, not all methods were designed with the same final tasks in mind, this will be compared in section 5.5. An discussion on the results is given in section 6.

5.1. Initial Observations

Some initial observations can be made by looking at Table 1. The first thing to notice is that most research is conducted on making the algorithms private. Specifically, they focus on adding noise to the training process of the generative model, like DPSGD [15], called Training Perturbation in the table. Furthermore, most focus on differential privacy. This can be explained by the fact that differential privacy provides rigorous privacy guarantees and is generally accepted by the research community [87]. An additional reason for the amount of research is that research done on privacy preserving machine learning does not only apply to synthetic data analysis and therefore has the potential of reaching a larger audience and having a larger research community. Possibly, variants of DPSGD could also be popular because it does not require distributed training. However, model privacy does not guarantee privacy in an MLaaS setting or in a collaborative setting. This is a big problem, because MLaaS can be helpful to make ML in research more accessible. With only privacy preserving models, one cannot safely generate synthetic data in the cloud and is therefore forced to do it locally.

Another thing to note is that the research on model privacy is mainly focused on GANs and not VAEs or other types of models. Especially more recently, the focus lies more heavily on GANs.

Recently, more research is done on making the training of machine learning models privacy preserving in the case of distributed data, i.e. in collaborative learning. More specifically, a part of the focus lies on differentially private federated learning. This technique can also be implemented in the data generation process, where the data is distributed across different data holders, who are reluctant to share the privacy sensitive data with each other or a central server.

Finally, since about 2019, techniques are being proposed that use the Rényi Differential Privacy Accountant [69] as opposed to the commonly used Moment Accountant by Abadi et al. [15], which could lead to an improved utility of the generated data because of the tighter bounds on privacy loss, meaning less noise is needed to offer the same amount of privacy guarantees.

5.2. Model Mechanisms

As mentioned in section 5.1, most research is focused on making the GAN training process differentially private. However, not all methods use GANs as their generative model, some use AE-based models or a mixture of different models. Therefore, in this section the methods are compared looking at what type of generative model they are designed for and specifically, the details of the implementation of their model. The obfuscate method by Zhang et al. [82] is not included in this section as it is not designed for a specific generative model, but is concerned with the input only.

5.2.1. GAN-Based Methods

A lot of methods are designed specifically for GANs. However, there are different design decisions to be made when implementing GANs. There are different frameworks for GANs, like the vanilla GAN, but also the Wasserstein GAN (WGAN) and the Improved Wasserstein GAN with Gradient Penalty (WGAN-GP) for example. The internal architecture of the generator and discriminator networks might also be different depending on the use case. Therefore, we'll compare the different GAN-based methods by framework, if they can also be easily adapted to other GAN frameworks, if they need public data and what architecture and implementation of the GAN they used, meaning the network architecture for discriminator and generator, the optimizer used and the activation functions in the nodes. An overview of this comparison can be found in Table 2.

All the methods focusing on GANs have one thing in common and that is that they exploit the Post-Processing theorem that allows them to only make the discriminator differentially private and therefore the generator as well.

By far most methods use the Wasserstein GAN (WGAN) or the WGAN with gradient penalty (WGAN-GP). These two are improvements on the vanilla GAN by providing better stability and convergence of training using the Wasserstein distance instead of the Jensen-Shannon divergence. GANs, especially when also adding noise to the gradients, are known for their training instability [51] and hence it might be beneficial to use the WGAN(-GP) instead of the vanilla GAN. A couple of methods use CGAN, but these are in minority.

Method	Framework	Other	Public Data Needed	D Network	G Network	Optimizer	Activation
dp-GAN	WGAN-GP	Yes	Yes	Similar to WGAN-GP	Similar to WGAN-GP	Adam	Similar to WGAN-GP
DPGAN	WGAN	Yes	No	Similar to DCGAN	Similar to DCGAN	RMSProp [94]	leaky ReLU for discriminator + ReLU for generator
FPGAN	DCGAN	N.A.	No	4-5 convolutional layers + linear classifier	2 linear layers + 5 deconvolution layers + fractional max pooling	Adam	leaky ReLU and sigmoid for discriminator + ReLU and tanh for generator
PATE-GAN	GAN	No	No	N.A.	N.A.	Adam	ReLU + Sigmoid for output
GAN-obfuscator	WGAN	No	Yes	Similar to DCGAN	Similar to DCGAN	AdaptRate	ReLU in generator + leaky ReLU in discriminator
EDP-GAN	WGAN-GP	N.A.	No	4 convolutional + fully connected linear + DP + linear classification layers	Fully connected linear layer + 3 deconvolution layers	Adam	SELU + tanh for last layer generator
DP-CGAN	CGAN	No	No	2 fully connected layers	2 fully connected layers	Adam for generator, (DP)SGD for discriminator	N.A.
cdp-GAN	WGAN-GP	Yes	No	Deep fully connected network	LSTM (stream) or MLP (discrete) with Softmax	Adam for generator, (DP)SGD for discriminator	N.A.
FL-GAN	WGAN-GP	N.A.	No	N.A.	N.A.	Adam	leaky ReLU
Asyn-DGAN	CGAN	Yes	No	PatchGAN [95] patch size 70x70	9-blocks ResNet [96]	Adam	ReLU

Table 2: GAN-based methods

Although they are designed and implemented with a specific framework in mind, most methods can be adopted into other GAN frameworks as well. The methods that do not work for other frameworks often explicitly mention this. Jordan et al. [85] explicitly mention that their method PATE-GAN needs to be extended to the regression setting to work with WGAN for example. GANobfuscator exploits the K-lipschitz property in WGAN and therefore needs to be changed to work with other frameworks. Finally, DP-CGAN was specifically designed for CGAN and is therefore not easily translatable to other frameworks without conditioning. Some methods do not specify whether it can be easily adapted to other frameworks, so these are marked with N.A. in Table 2.

Furthermore, most do not require the use of (a small set of) public data to work, the exceptions being dp-GAN and GANobfuscator that need a small set of public data for optimizations as explained in section 4. They both use it for an optimization that adaptively determines the clipping/pruning bounds and dp-GAN also uses it for the warm starting strategy, where it uses the public data to initialize the model before training it on private data. More on the reliance on public data in section 5.4.

The network architectures used for the discriminator and generator defer quite a bit and not all methods explicitly mention the architecture that they used for the discriminator and generator. Most use fully connected networks or convolutional neural networks with different levels of depth and number of nodes. However, cdp-GAN uses Long Term Short Memories for stream data and a Multilayer Perceptron with a softmax layer for discrete data in the generator. Triastcyn et al. mention that FPGAN only works for feed-forward networks [81].

Some methods mention that their implementation has an architecture similar to the DCGAN architecture that removes fully connected layers for deep architectures and uses batch normalization and convolutions among other things [53].

Furthermore, most use the Adam optimizer, which is an extension of stochastic gradient descent [97]. The main difference to regular SGD is that Adam computes separate adaptive learning rates for different parameters in the model using both the mean and the variance of the gradients [97]. DPGAN and GANobfuscator use RMSProp[94] and AdaptRate [87] respectively as opposed to Adam. AdaptRate and RMSProp are both optimization algorithms that adaptively adjust the learning rate according to the magnitude of the gradients.

Finally, leaky ReLU as activation function in the discriminator and ReLU in the generator are the most often used. Some methods use different activation functions for the output layer, like tanh in the generator and sigmoid in the discriminator and EDP-GAN uses SELU [98] as activation function instead of (leaky) ReLU.

To summarize, there is quite some diversity in how the methods are implemented and designed, even if they have GAN in common. This is mainly because there are a lot of different frameworks for GANs and a lot of possible architectures, one better suited for a certain use-case than others.

5.2.2. AE-Based Methods

Some methods focus on AE-based models, like the AE, the deep AE and the VAE. They differ in the architecture of the AE model, whether they need public data and how they eventually sample the data, which can be seen in Table 3.

The dPA is the only one that focuses on the deep AE, which means multiple different AEs stacked on top of each other. DP-SYN and DP-AuGM focus on the AE and DP-VaeGM and PriVAE focus on the VAE. Although DP-SYN and DP-AuGM both focus on the AE, they do not work the same. DP-AuGM needs public data to feed into the encoder to sample the data, while the DP-SYN uses the decoder for sampling and therefore uses a differentially private algorithm called DP-EM [91] to take samples from the latent space that are then decoded into new data, which does not need public data. PriVAE by Takahashi et al. is one of the methods that focuses on making VAEs differentially private in an optimal way without using them in ensembles. However, neither PriVAE nor dPA focus on sampling, but focus on other tasks, more on that in section 5.5. DP-VaeGM in turn feeds samples from a Gaussian distribution into the model to create new samples.

All methods, except dPA, use SGD for training the (V)AE. To make SGD differentially private, DP-SYN, DP-AuGM and Dp-VaeGM use something similar to DPSGD, where PriVAE made an adjustment to decrease the sensitivity. dPA does something similar to DPSGD, adding noise to the polynomial coefficients of the objective function during training.

To summarize, most differences are in which autoencoder version specifically is used, deep AE, AE or VAE, but also in how they generate the data, since some use the decoder and others the encoder as generative model and therefore need to feed it different types of data (public data versus drawn from distribution). While others are not focused on generating data at all.

5.2.3. Ensembles

Finally, some of the methods use multiple models during training. They train these models on disjoint datasets and eventually combine the results of these models to create a new synthetic dataset. The way these methods partition the data, divide this data across the different models and then eventually combine it again, is different per method. An overview is given in Table 4. Some of the methods in this table are also mentioned in sections 5.2.1 and 5.2.2, because they either use GANs or AEs in their ensembles.

One thing these methods have in common is that they all do not need public data as opposed to some of the other methods in sections 5.2.1 and 5.2.2.

The way this data is partitioned differs however. DPGM uses a private version of the K-Means clustering algorithm to determine clusters of the dataset that contain samples that are similar. DP-SYN and DP-VaeGM divide by labels, where each model then receives data corresponding to one label. These three methods do have in common that they try to assign data to models that are similar to each other, either through clusters or the same labels. This way the models are specialized in one type of group in the data. PATE-GAN on the other hand divides the data into k even subsets without taking into account similarity and then assigns these datasets over the different teachers. Finally, since FL-GAN and AsynDGAN are solutions for the collaborative learning setting, where the data is already partitioned, the models are located where the data is.

Furthermore, to combine these models and then generate data, DPGM mimics the training process, randomly selecting one of the models to generate one sample depending on how often they were selected during training, whereas DP-SYN simply appends the data of the different models and DP-VaeGM takes the union of the data of the different models.

DP-SYN compares itself extensively with DPGM. The data generated by DPGM performs better on classification tasks on certain datasets (BreastCancer and Diabetes), but performs worse on most of the others. DP-SYN performs better on imbalanced or high dimensional datasets. This is especially true for low values of epsilon.

PATE-GAN does not need to combine the models after training as they are combined during training into the student discriminator and the single generator. PATE-GAN uses something different as it is an implementation of PATE. The trained teacher discriminators receive generated samples from the generator and then they each give a vote on the classification of the samples. These votes are aggregated into one vote, where also noise is added. This noisy vote is then passed to the student discriminator and eventually through this the generator is trained. Not much research has been done on trying to integrate PATE in the generative models for synthetic data and PATE-GAN is one of the few implementations focusing specifically on exploiting PATE for privacy preserving data synthesis [85]. The authors compare PATE-GAN extensively with DPGAN and the results

Method	Version	Public Data Needed	Architecture	Sampling
dPA	deep AE	No	2 private AEs + softmax layer	N.A.
DP-SYN	AE	No	N.A.	DP-EM [91]
DP-AuGM	AE	Yes	N.A.	Feed public data to encoder
DP-VaeGM	VAE	No	N.A.	Feed samples from Gaussian distribution to model
PriVAE	VAE	No	Convolutional neural networks for sparsity + fully connected neural network for clustering	N.A.

Table 3: AE-based methods

Method	Models	Partitioning	Combining
DPGM	VAE + RBM	Private K-Means Clustering	Random sampling
DP-SYN	AE	Labels	Appending
PATE-GAN	GAN	k even subsets	Teacher vote aggregation
DP-VaeGM	VAE	Labels	Taking the union
FL-GAN	GAN	Data location	Serial training
Asyn-DGAN	GAN	Data location	Central generator

Table 4: Ensemble-based methods

of their experiments are that PATE-GAN performs better for every value of epsilon. This mimics the results by Papernot et al., who claim that PATE performs better than DPSGD [74].

FL-GAN and AsynDGAN both implement a variant of centralized FL. Therefore, both have a central server. FL-GAN uses this central server only as orchestrator of the process. The server initializes the model, sends this to one client, who trains on it and returns the new model. The server then sends this new model to another client and the process terminates after all clients have trained their local data on the model. This is called serial training as it is performed in a synchronous manner, a new client starts training after the other completed their part. Then after the training is finished one single model is the result. So no need to combine anything after. Xin et al. compare their method with regular federated learning and differentially private federated learning. This shows that their method is comparable to regular federated learning and better than differentially private federated learning regarding image quality.

The same goes for AsynDGAN that uses the central server as an aggregator. The server acts as the generator in the GAN. All the individual discriminators receive data from the generator and calculate the loss and this information is returned to the server for each discriminator, where this loss is aggregated and used to update the generator. This results in one single generator model as well.

So to summarize, there are multiple flavours of ensembling multiple models. One of them is centralized federated learning, where multiple participants and one central server collaborate to train one final generator model. Another is PATE, where multiple teacher models are used to train a student model that in the case of PATE-GAN trains one single generator. Finally, you have methods that partition the data by looking at similarities and assign this to different models. These methods require a more complicated generation strategy as the results and data of the different models need to be combined after training, because the training process produces multiple different models.

5.3. Privacy Mechanisms

A lot of methods enforce some form of differential privacy. In this section, the different ways this is applied are compared. An overview of this can be found in Table 5. In this table, one can see the type of perturbation used, i.e. input perturbation, where the input is perturbed before feeding it into the model, model perturbation, where parameters of the model itself are perturbed in some way during training and output perturbation, where the output of the model is perturbed. This output perturbation can also happen during training, where the output of the discriminator in GAN is perturbed that is communicated with the generator for example, but the difference is that it does not perturb the parameters of the model. The discriminator is not trained in a differentially private way, but the output is differentially private and therefore the generator is. In Table 5, one can also see which version of differential privacy is enforced and how, i.e. what noise mechanism is used, where is the noise added and how do they keep track of the total privacy loss. The table only shows privacy through perturbation. As can be seen in the overview table (Table 1), there are also some other methods like federated learning.

Most methods use model perturbation by adding Gaussian noise to the gradients during gradient descent in the backward pass. dPA is the only one that enforces ϵ-differential privacy and uses the Laplacian noise mechanism instead of the Gaussian noise mechanism used by the others. Furthermore, it does not add noise to the gradients during gradient descent but to

Method	Type	DP	Noise	Location	Accounting
dPA	Model	Epsilon	Laplacian	Polynomial coefficients of objective function	Normalization layer before stacking AEs
dp-GAN	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant
DPGAN	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant
FP-GAN	Model	Epsilon, Delta	Gaussian	Forward Pass	Moments Accountant
Obfuscate	Input	N.A.	Gaussian	Individual sample	None
DPGM	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant
DP-SYN	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant
PATE-GAN	Output	Epsilon, Delta	Gaussian	Teacher vote	Moments Accountant
DP-AuGM	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant
DP-VaeGM	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant
GANobfuscator	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant
EDP-GAN	Output	Epsilon, Delta	Gaussian	Discriminator output	Ex-post analysis by KL-divergence estimator + Chebyshev's inequality
DP-CGAN	Model	Rényi	Gaussian	(SGD) Gradient	Rényi Differential Privacy Accountant
cdp-GAN	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant
FL-GAN	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant
AsynDGAN	N.A.	N.A.	None	N.A.	None
PriVAE	Model	Epsilon, Delta	Gaussian	(SGD) Gradient	Moments Accountant

Table 5: Privacy through perturbation

the polynomial coefficients of the objective function. Since dPA enforces ϵ-differential privacy, it provides stricter privacy guarantees, because it does not use one of the relaxations of ϵ-differential privacy, like (ϵ, δ)-differential privacy. In general, ϵ-differential privacy is perceived as too rigid. Therefore, all the other methods use a relaxation of ϵ-differential privacy. They use Gaussian instead of Laplacian noise, because where the Laplacian mechanism satisfies ϵ-differential privacy, the Gaussian mechanism satisfies (ϵ, δ)-differential privacy. DP-CGAN is the only one that enforces the relatively new Rényi Differential Privacy. All the other methods that enforce DP enforce (ϵ, δ)-differential privacy.

One of the first methods to implement DPSGD to enforce (ϵ, δ)-differential privacy in GAN for data synthesis was DP-GAN by Xie et al. [80]. The results were promising, but the used epsilon was quite high (higher than 9) and the degradation of the data quality was severe even with high epsilon. So, to get fairly good data quality one needs epsilon values that are even higher, decreasing privacy. Therefore, improvements or different strategies were needed to be able to achieve higher quality when using GANs in a differentially private way to generate data.

Not all methods use a variant of DPSGD. FP-GAN is one such method [81]. One downside of FP-GAN is that it only works for feed-forward neural networks, therefore it does not work on recurrent neural networks. The experiments performed by Triastcyn et al. focus on the results from Papernot et al (PATE) and it shows that their approach performs slightly better (by 0.09%) than PATE, but much worse on SVHN (7.16%). They attribute this to the more generic nature of their approach. They do not compare their method privacy-wise. So, it is unclear whether the privacy is also worse because of the more generic nature.

Another method without DPSGD, EDP-GAN was also proposed by Triastcyn et al. [88]. The problem with this method is that it has no proven strict privacy guarantees and therefore the privacy of EDP-GAN needs to be further checked. It does perform better than PATE, but this could be due to the fact that it has less rigorous privacy guarantees. To show that their method does protect privacy, they perform some attacks on their model and this is promising. However, formal proofs are not provided.

There is a newer relaxation of differential privacy that provides stricter privacy guarantees than (ϵ, δ)-differential privacy, but is more relaxed than ϵ-differential privacy, called Rényi Differential Privacy, which is used by DP-CGAN [44]. RDP provides stricter privacy guarantees, so would be preferable if the utility is preserved. Since it is a fairly new kind of differential privacy, it has only one implementation in the data generation process. Torkzadehmahan et al. test the quality of the synthetic data on regular conditional GAN (CGAN). The results on quality of the data is that the synthetic data with RDP is about 5% worse than that of the regular CGAN. However, when regular DP is added to the CGAN, DP-CGAN performs 5% better. Another benefit of DP-CGAN is that it generates labels as well as data and the synthetic data therefore easily be used for supervised learning tasks without the need for labeling.

Obfuscate is the only one that focuses on input perturbation. Instead of adding noise to the model parameters or to the output, it adds noise to the individual samples before passing it on to the model. Obfuscate also protects the privacy of groups of

samples by adding new samples. It does not guarantee differential privacy and also does not use privacy accounting to keep track of the privacy loss. AsynDGAN is another method that does not do use differential privacy, but as opposed to Obfuscate AsynDGAN does not perturb anything at all, relying only on federated learning.

Both PATE-GAN and EDP-GAN use output perturbation (during training) instead of model perturbation. With PATE-GAN the teacher votes are made noisy before being send to the student discriminator and with EDP-GAN the output of the discriminator is perturbed. FP-GAN uses model perturbation but instead of adding noise to the gradients during gradient descent in backward pass, it is added to forward pass by adding a Gaussian noise layer to the discriminator.

Finally, most methods use the Moments Accountant. However, dPA uses a normalization layer before stacking AES, but no other type of accounting. This normalization layer guarantees DP before moving on to next layer. DP-CGAN uses the Rényi Differential Privacy Accountant, because it tries to enforces Rényi Differential Privacy and EDP-GAN uses an ex post analysis using KL-divergence estimator and Chebyshev's inequality to determine privacy bound.

To summarize, the majority of methods enforce (ϵ, δ)- differential privacy by adding Gaussian noise to the gradients during gradient descent in the backward pass when training the discriminator and keep track of the privacy budget by using the Moments Accountant. There are a couple of methods that deviate from this technique by using different noise, enforcing another type of differential privacy, use a different type of perturbation or use another accounting method.

5.4. Reliance On Public Data

GANs are the more popular model for generating data compared to VAEs, although they suffer from training instabilities [51]. To improve this, some of the GAN methods add a small portion of public data to the training process to perform some optimization techniques. This has already been touched upon in section 5.2.1 and it is shown in Table 2.

dp-GAN is one of the methods that needs public data [79]. This public data is used during adaptive clipping, which leads to faster convergence and higher utility. dp-GAN is able to generate synthetic data with Inception scores and Jensen-Shannon divergence close to that of the real data. It performs slightly worse than that of regular GAN, but this is because of the added noise during training. DPGAN was published in the same year, but that method does not need public data and its performance was promising, but not very usable, requiring large values of epsilon to work. That is the trade-off between using public data or not. GANobfuscator is very similar to dp-GAN [87]. It needs public data as well, and GANobfuscator does perform better in both the quality and privacy experiments than both DPGAN and dp-GAN looking at the Inception Scores and the Jensen-Shannon Scores and performing a membership inference attack. The data generated by GANobfuscator even performs better when being used to train a classifier than that of the regular GAN, although the two do approach each other the more training samples are used.

There is one method that also uses public data, but does not use it for optimizations and is AE-based. DP-AuGM by Chen et al. uses the public data for generating samples and compared to DP-VaeGM, also by Chen et al. [86], this public data results in a more stable data quality than with the Gaussian noise used in DP-VaeGM. It provides a fairly good prediction accuracy on all tested datasets for even small epsilon (i.e. 1). Most importantly, the accuracy increases with the use of the generated data alongside the real data compared to using only the real data. It performs better than plain DPSGD and the scalable version of PATE. Furthermore, it is shown to be effective against Model Inversion, Membership Inference and the GAN-based Attack, while DPSGD, PATE and DP-VaeGM only protect against Membership Inference. However, this is at the cost of needing public data, but if this data is available that is not a problem.

cdp-GAN is an adjustment to dp-GAN to allow for the generation of time series, continuous and discrete data [89]. The strategy of clipping decay implemented in cdp-GAN improves the dp-GAN accuracy by about 1 to 2% and very closely approximates that of GAN, while it does not require public data at all, which is an indication that one not always need public data to get good results.

5.5. Application

Not all proposed models are focused on data generation. Some are focused on other tasks like classification. This is because model privacy is linked to the research of privacy preserving machine learning (PPML). PPML does not only focus on data generation however, but all tasks for machine learning. Most methods do use one of the popular dataset, like MNIST or CelebA. An overview of the task the methods were designed for and on what datasets they were tested is shown in Table 6.

One of the methods that was not designed specifically for data synthesis is dPA. It is designed specifically for human behavior prediction. Furthermore, PriVAE is only tested on a sparse coding and clustering task instead of on data synthesis. The method performs well on the clustering task, where it is able to cluster a spinwheel shaped dataset correctly. Furthermore, Takahashi et al. managed to increase the sparsity even under DP-constraints. Although they do say that PriVAE needs to be improved on its reconstruction performance to increase the sparsity even more. Obfuscate was demonstrated with a classification use-case, but since Obfuscate does not depend on the type of model or what happens after the input is perturbed, it can also be used for generative model training. The most common use-case of the methods is synthesis however, as that is the focus of this paper and these methods were therefore selected accordingly.

The synthesis is all mostly concerned with images, trying to mimic the MNIST or CelebA datasets for example. GANs are often used for image synthesis, but can definitely be used for other types of data as well, just like cdp-GAN showed.

Method	Data	Task
dPA	Health Social Network Data	Human Behavior Prediction
dp-GAN	MNIST, LSUN, CelebA	Synthesis
DPGAN	MNIST, MIMIC-III	Synthesis
FP-GAN	MNIST, SVHN	Synthesis
Obfuscate	CIFAR10, AT&T Face Data, MNIST, EMNIST	Classif-ication
DPGM	MNIST, CDR, TRAN-SIT	Synthesis
DP-SYN	Many i.e. ODR, CMC	Synthesis
PATE-GAN	Many i.e. MAGGIC	Synthesis
DP-AuGM	Many i.e. MNIST	Synthesis
DP-VaeGM	Many i.e. MNIST	Synthesis
GANobfuscator	MNIST, LSUN, CelebA	Synthesis
EDP-GAN	MNIST, SVHN, CelebA	Synthesis
DP-CGAN	MNIST	Synthesis
cdp-GAN	IoT City of Antibes, UCI Adult Data	Synthesis
FL-GAN	MNIST, CelebA	Synthesis
AsynDGAN	BraTS2018, Multi-Organ	Synthesis
PriVAE	Fashion-MNIST	Sparse Coding, Clustering

Table 6: Application domain

6. Discussion

After the comparison in section 5, this section will provide a more conceptually inclined discussion on the results, structured according to topic and challenge. Hopefully, this provides insights on characteristics, strengths and shortcomings of the current methods, paving the way to better understand the state-of-the-art and plan future research.

6.1. Assumptions

A few common assumptions are made by the authors of the methods discussed in this paper. One of them is the availability of public data. These authors say that it is a reasonable assumption that there will always be some public data available to help models be trained for their specific purpose. However, this is not always the case and hence that this assumption is not always correct. In cases where public data is not available, one still needs privacy protection when generating data and these methods will not be of use. Therefore, methods that do not need public data need to be cultivated and developed. However, in the case that there is public data available, these methods often provide better training stability and data utility. PATE-based methods usually need public data as well, but PATE-GAN variant avoids that because Jordan et al. [85] also find it an unreasonable assumption.

Furthermore, this public data is often used for optimizations to improve training stability and data utility. However, public data should not be a standard to require for privacy preserving methods because, as cdp-GAN showed, it's not always necessary where cdp-GAN performs better than dp-GAN without the need of public data for the optimizations. Optimizations are needed however and should not be omitted entirely to avoid public data. But they could be redesigned, because regardless the GAN frameworks are known for their training stability issues. Hyperparameters are shown to impact the performance of generative models quite severely and forms of hyperparameter tuning could help improve performance.

Another point is that most methods do no explicitly mention whether they take into account bias in the data or that the data is skewed in some way. It is also possible that the data is not equally distributed among different parties and to our knowledge this is not explicitly addressed and dealt with. Furthermore, the methods are not tested on how they perform with different sizes of data, e.g. in presence of small datasets in a collaborative data generation setting. Only some of the ensemble methods explain that multiple models can be used to create a more balanced dataset by sampling more or less from a certain class or cluster.

Finally, all the methods concerned with model privacy assume that the place where the data is stored and the model is run is trusted. This is not the case with MLaaS, rendering these methods useless in that setting. Even a method such as AsynDGAN which is concerned with collaborative privacy, assumes that the central server can be trusted as it exchanges information on the model with this server without using any kind of perturbation or cryptographic protection. If the server is compromised, an adversary could potentially use this information exchange to leak information on the training data.

6.2. Privacy Mechanisms

Most methods use (ϵ, δ)-differential privacy to design and fine tune their perturbation mechanism. However, other interpretations of differential privacy, e.g. Rényi Differential Privacy (RDP), have shown promise. Sadly, no comparison is done with DP-CGAN (the only method implementing RDP) and other DP GANs. So, it is unknown the effect of privacy regime on privacy budget spending and privacy bounds of their model compared to that of vanilla CGAN (conditional GAN). It would be interesting to see if the use of RDP and RDP accounting improves privacy and utility. There are other types of differential privacy as well that have no implementation in PPSGD to our knowledge, for example Gaussian Differential Privacy [71].

A current problem with privacy accounting is that the bound is worst-case, meaning that the actual privacy loss might be better. A tighter privacy is always desirable to allow for less noise to be added and hence better utility. As an example, this is where the RDP (accountant) might be a real improvement, but this needs further research.

6.3. Model privacy for PPSDG

Making models private basically comes in two flavours: perturbation methods that add noise somewhere during training,

like DPSGD, and ensemble methods to train (parts of) models on disjoint datasets, like PATE. By far most research in PPSGD is focused on this area.

The research on using DPSGD already has lots of contributions, especially when it comes to GANs (VAE less much so, which could still be interesting). In general, either (variational) auto-encoders or generative adversarial networks are studied. However, no real research has been done on PATE in either GAN or VAE, although there are some other ensemble methods proposed.

Some lessons can be learned however from looking at these ensemble methods. One of them is that using multiple models is promising as it might help utility. This is because models specialize in a specific part of the dataset that are very similar. For example, one model would specialize in generating horses, the other in cars. And because one would split them up instead of training one single model, one would not accidentally get an image that looks like a mix between a horse and a car. Furthermore, multiple models can also be used to create a more balanced dataset by sampling more or less from a certain class or cluster. Most of these tested combination approaches use AEs or VAEs though. So, it might be interesting to see more examples of how it works with GANs.

Finally, Takahashi et al. [90] came up with a new version of DPSGD called term-wise DPSGD that can also be used in other deep learning models to satisfy differential privacy and it would therefore be interesting to see how it affects the performance of GANs, because GANs suffer from training instability and this might improve both training stability and data quality.

6.4. Privacy in Collaborative Learning

None of the methods implementing either DPSGD, PATE or similar focus on the scenarios where the data is distributed among different users. In this collaborative learning setting, it is important that the privacy of the individual datasets is preserved.

The two methods explicitly focusing on the collaborative learning setting tried to implement some form of federated learning. An issue with one of the approaches, AsynDGAN, is that the authors do not provide any rigorous privacy guarantees, not acknowledging the fact that the information shared between discriminator and generator can potentially leak information that can be exploited by, for example, GAN-based attacks [59]. They assume that by only communicating auxiliary data, fake data and discriminator loss the privacy is guaranteed. However, Hitaj et al. [59] showed that all collaborative solutions are inherently vulnerable. To be completely sure about the privacy guarantees some experiments and proofs are needed. The approach taken by the authors is interesting however and if more guarantees on privacy can be given, further research is definitely interesting.

Theoretically speaking all ensemble methods can be used in the collaborative learning setting with some small changes as they are inherently distributed. However, the way the data is divided might be different. As with collaborative learning the data is already divided, thus there is no need for k-means clustering or label-division.

6.5. Input Privacy for MLaaS

In the MLaaS setting, it is imperative that users can safely and privately run the generation process in the cloud without providing the service owner with the unprotected training data. Not many methods take this scenario into account and hence do not provide privacy in the case where there is no trusted third party to run the generation process on. Note that even the solutions under collaborative privacy assume some form of a trusted central server to act as the central generator or the delegate of the entire process.

Surprisingly, the research on privacy preserving data generation in the case of MLaaS has not received any big contributions. Our expectation is that the usage of MLaaS will rise, because it offers the convenience of machine learning without the need of having extensive resources or skills to use it.

Chen et al. claim that their method works with MLaaS, but they focus on synthetic data as the solution [86]. They say that one can generate synthetic data using their techniques, which requires little resources, and then send this data to the MLaaS provider. However, as mentioned in section 3.1, when one wants to generate the data on the MLaaS server, this obviously does not work and is not a solution to the problem as we stated, namely safe generation of data on MLaaS.

Zhang et al. [82] are one of the few that address the problem explicitly by using an Obfuscate layer. Experiments show that the method is effective against model memorization, membership inference, model inversion and model classification attacks. However, no differential privacy is used or and no other rigorous privacy proofs are provided and therefore the privacy cannot be guaranteed.

6.6. Policy

Lastly we would like to touch upon matters related to policies and regulations governing the privacy of individual data. Privacy regimes like GDPR have been a leap forward in protecting the privacy of the individuals in terms of data collection, data access, analysis and aggregation, etc. With raising awareness about privacy, local policies are being put in place as well by institutes and companies that act to further protect the privacy of the individuals, but also make access and usage of the data more complicated. But one gray area is the use of data products after transformations and aggregations inherent in machine learning pipelines, especially when we deal with collaborative learning and introduce synthetic data to the equation. Consider PATE-GAN, an architecture built on the concept of PATE employing synthetic data generation. Data resides in different private datasets, either individual or data silos. The data is used to train teacher models, which then in turn will be used alongside synthetic data to train a student model. Even though the data generation model is privacy preserving, along the pipeline the data from different generators will be accumulated, then fed to another machine learning mechanism. Given that it is highly probable that in real world different private datasets belong to different institutions, each with their own data protection policies, policies governing the end product of such a system fall into a gray area. To the best of our knowledge there is not any legal regime present that would readily

and clearly address the usage of higher degree data products. It will be simplistic to assume that intersection of all the policies will be considered since putting aside the fact that the compatibility of different regulation frameworks is highly debatable, even if different policies are translatable and can be aligned, intersection of them would probably be too strict to have any real utility.

This area of research is still open and in dire need of attention and development. Considering the fact that each privacy preserving method has its own caveats and utility concerns usually force us to accept a trade off, and keeping in mind that it is simplistic to assume that any implementation of these methods will be flawless, we argue that a policy regime is needed to not only guide researchers and designers to better tailor their pipelines to maximize privacy but also to be the reference in the probable event of a privacy breach.

7. Directions for Future Research

During the discussion in section 6, some possible interesting takes on research were already mentioned. In this section we provide a general overview of the areas in privacy preserving synthetic data generation that might benefit from further research.

7.1. Alternative Differential Privacy Regimes

(ϵ, δ)-differential privacy is the most popular version of differential privacy enforced in privacy preserving synthetic data generation. A problem however is that the privacy bound is often too tight. Other differential privacy variants have been proposed that could potentially help with this, providing a more accurate privacy guarantee without the need for as much noise, but have not been tested on PPSGD yet.

7.2. Collaborative Learning

Another challenge in synthetic data generation is the case of collaborative learning. Often, data is distributed across different entities that want to collaboratively generate synthetic data, but do not or cannot share their data with each other. Federated learning is a technique that can be used, but it does not provide rigorous privacy guarantees. Furthermore, some ensemble methods used for training data from one single user, are inherently distributed and could be adopted in a collaborative learning setting.

7.3. Privacy in MLaaS

Most methods assume that the location of the data and the model is trusted. If this is not the case, these methods do not preserve privacy as the data is stored and the model is trained at this untrusted location. Not much research is conducted on making synthetic data generation methods private in this setting as well. Since the use of MLaaS could potentially grow in the research community due to its easiness to use, as it makes machine learning more available to people, who do not have access to expensive resources to run the models, it is imperative to investigate methods to make it private. Cryptographic methods, specifically homomorphic encryption, could be useful here if integrated efficiently in the generation process as they allow the users to encrypt their data before sending it to the MLaaS provider. To our knowledge this has not received much research yet.

7.4. Empirical Testing

The ways privacy is tested and determined differs significantly from method to method. Some do not even provide experiments on privacy. To standardize this and make it more easy is some from of automatic empirical testing on methods, so one can easily check the privacy guarantees of their method.

7.5. Real-World Data

Often there is no mention of the type and quantity of data used or whether it is evenly distributed. The effects of using data with different kinds of characteristics in that regard are unknown, whereas often real-world data is not the same as the neatly formed data used for testing.

7.6. Privacy Policies and Regulations

No research has been done on the implications of privacy policies and regulations regarding individual data. With the increasing awareness of the importance of privacy, more policies and regulations are put in place. How these policies and regulations affect synthetic data are unclear. Therefore, policy regimes are needed to create clarity, especially in the case of a data breach.

8. Conclusion

Synthetic data has the potential to resolve some big problems in data intensive use-cases. However, privacy concerns over synthetic data are as real and serious as over real data. In this paper, we looked at different methods of machine learning-based privacy preserving synthetic data generation. We provided detailed information on the methods and analyzed these methods both in depth and from a high level viewpoint. We also gave an overview of research gaps and what still needs to be done. This paper aims to act as a reference for researchers in this area and incite future research on developing PPSDG methods and their utilization into greater privacy preserving machine learning schemes.

9. Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

10. Acknowledgement

This research has been performed as part of the *Enabling Personalized Intervention* (EPI) project. The EPI project is funded by the Dutch Science Foundation in the Commit2Data program, grant number 628.011.028.

References

[1] T. Zhou, S. Ruan, S. Canu, A review: Deep learning for medical image segmentation using multi-modality fusion, Array 3 (2019) 100004.

[2] X. Liu, L. Faes, A. U. Kale, S. K. Wagner, D. J. Fu, A. Bruynseels, T. Mahendiran, G. Moraes, M. Shamdas, C. Kern, et al., A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis, The lancet digital health 1 (6) (2019) e271–e297.

[3] R. A. Khalil, E. Jones, M. I. Babar, T. Jan, M. H. Zafar, T. Alhussain, Speech emotion recognition using deep learning techniques: A review, IEEE Access 7 (2019) 117327–117345.

[4] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, K. Shaalan, Speech recognition using deep neural networks: A systematic review, IEEE Access 7 (2019) 19143–19165.

[5] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, K. Dietmayer, Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges, IEEE Transactions on Intelligent Transportation Systems.

[6] S. Grigorescu, B. Trasnea, T. Cocias, G. Macesanu, A survey of deep learning techniques for autonomous driving, Journal of Field Robotics 37 (3) (2020) 362–386.

[7] P. Voigt, A. Von dem Bussche, The eu general data protection regulation (gdpr), A Practical Guide, 1st Ed., Cham: Springer International Publishing.

[8] P. Edemekong, P. Annamaraju, M. Haydel, Health insurance portability and accountability act, StatPearls.

[9] L. Sweeney, k-anonymity: A model for protecting privacy, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10 (05) (2002) 557–570.

[10] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.

[11] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 1322–1333.

[12] C. Dwork, A. Roth, et al., The algorithmic foundations of differential privacy., Foundations and Trends in Theoretical Computer Science 9 (3-4) (2014) 211–407.

[13] A. Friedman, A. Schuster, Data mining with differential privacy, in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 493–502.

[14] O. Williams, F. McSherry, Probabilistic inference and differential privacy, in: Advances in Neural Information Processing Systems, 2010, pp. 2451–2459.

[15] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 308–318.

[16] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al., Privacy-preserving deep learning via additively homomorphic encryption, IEEE Transactions on Information Forensics and Security 13 (5) (2017) 1333–1345.

[17] J. H. Cheon, D. Kim, Y. Kim, Y. Song, Ensemble method for privacy-preserving logistic regression based on homomorphic encryption, IEEE Access 6 (2018) 46938–46948.

[18] T. Zhang, T. Zhu, P. Xiong, H. Huo, Z. Tari, W. Zhou, Correlated differential privacy: feature selection in machine learning, IEEE Transactions on Industrial Informatics 16 (3) (2019) 2115–2124.

[19] R. C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: A client level perspective, arXiv preprint arXiv:1712.07557.

[20] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, Y. Zhou, A hybrid approach to privacy-preserving federated learning, in: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, 2019, pp. 1–11.

[21] M. Seif, R. Tandon, M. Li, Wireless federated learning with local differential privacy, arXiv preprint arXiv:2002.05151.

[22] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, A. Das, Differential privacy-enabled federated learning for sensitive health data, arXiv preprint arXiv:1910.02578.

[23] R. Shokri, V. Shmatikov, Privacy-preserving deep learning, in: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1310–1321.

[24] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, T. Goldstein, Poison frogs! targeted clean-label poisoning attacks on neural networks, in: Advances in Neural Information Processing Systems, 2018, pp. 6103–6113.

[25] Y. Zhang, C. Ling, A strategy to apply machine learning to small datasets in materials science, Npj Computational Materials 4 (1) (2018) 1–8.

[26] H. Lee, S. Yune, M. Mansouri, M. Kim, S. H. Tajmir, C. E. Guerrier, S. A. Ebert, S. R. Pomerantz, J. M. Romero, S. Kamalian, et al., An explainable deep-learning algorithm for the detection of acute intracranial haemorrhage from small datasets, Nature Biomedical Engineering 3 (3) (2019) 173.

[27] E. D. Cubuk, A. D. Sendek, E. J. Reed, Screening billions of candidates for solid lithium-ion conductors: A transfer learning approach for small data, The Journal of chemical physics 150 (21) (2019) 214701.

[28] Q. Sun, Y. Liu, T.-S. Chua, B. Schiele, Meta-transfer learning for few-shot learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 403–412.

[29] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, M. Cord, Boosting few-shot visual learning with self-supervision, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8059–8068.

[30] J. Zhang, C. Zhao, B. Ni, M. Xu, X. Yang, Variational few-shot learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1685–1694.

[31] D. B. Rubin, Statistical disclosure limitation, Journal of official Statistics 9 (2) (1993) 461–468.

[32] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, S. Birchfield, Training deep networks with synthetic data: Bridging the reality gap by domain randomization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2018.

[33] Q. Wang, J. Gao, W. Lin, Y. Yuan, Learning from synthetic data for crowd counting in the wild, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[34] D. Ward, P. Moghadam, N. Hudson, Deep leaf segmentation using synthetic data (2019). `arXiv:1807.10931`.

[35] Z. Tang, M. Naphade, S. Birchfield, J. Tremblay, W. Hodge, R. Kumar, S. Wang, X. Yang, Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[36] R. Feng, J. Gu, Y. Qiao, C. Dong, Suppressing model overfitting for image super-resolution networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2019.

[37] T. Sutojo, A. Syukur, S. Rustad, G. F. Shidik, H. A. Santoso, P. Purwanto, M. Muljono, Investigating the impact of synthetic data distribution on the performance of regression models to overcome small dataset problems, in: 2020 International Seminar on Application for Technology of Information and Communication (iSemantic), IEEE, 2020, pp. 125–130.

[38] H. Surendra, H. Mohan, A review of synthetic data generation methods for privacy preserving data publishing, International Journal of Scientific & Technology Research 6 (3) (2017) 95–101.

[39] B. Lubarsky, Re-identification of "anonymized data", UCLA L. REV 1754 (2010).

[40] A. Narayanan, V. Shmatikov, How to break anonymity of the netflix prize dataset, arXiv preprint cs/0610105.

[41] P. Regulation, Regulation (eu) 2016/679 of the european parliament and of the council, REGULATION (EU) 679 (2016) 2016.

[42] A. Yale, S. Dash, R. Dutta, I. Guyon, A. Pavao, K. P. Bennett, Generation and evaluation of privacy preserving synthetic health data, Neurocomputing.

[43] E. L. Barse, H. Kvarnstrom, E. Jonsson, Synthesizing test data for fraud detection systems, in: 19th Annual Computer Security Applications Conference, 2003. Proceedings., IEEE, 2003, pp. 384–394.

[44] R. Torkzadehmahani, P. Kairouz, B. Paten, Dp-cgan: Differentially private synthetic data and label generation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Ad-

vances in neural information processing systems, 2014, pp. 2672–2680.
[46] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114.
[47] F. Rosenblatt, Principles of neurodynamics. perceptrons and the theory of brain mechanisms, Tech. rep., Cornell Aeronautical Lab Inc Buffalo NY (1961).
[48] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural computation 1 (4) (1989) 541–551.
[49] J. Schmidhuber, Deep learning in neural networks: An overview, Neural networks 61 (2015) 85–117.
[50] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein gan, arXiv preprint arXiv:1701.07875.
[51] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. C. Courville, Improved training of wasserstein gans, in: Advances in neural information processing systems, 2017, pp. 5767–5777.
[52] M. Mirza, S. Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784.
[53] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434.
[54] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning internal representations by error propagation, Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science (1985).
[55] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F. E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.
[56] D. P. Kingma, M. Welling, An introduction to variational autoencoders, arXiv preprint arXiv:1906.02691.
[57] A. Dandekar, R. A. Zen, S. Bressan, Comparative evaluation of synthetic data generation methods, in: Proceedings of ACM Conference (Deep Learning Security Workshop), 2017.
[58] M. Al-Rubaie, J. M. Chang, Privacy-preserving machine learning: Threats and solutions, IEEE Security & Privacy 17 (2) (2019) 49–58.
[59] B. Hitaj, G. Ateniese, F. Perez-Cruz, Deep models under the gan: information leakage from collaborative deep learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 603–618.
[60] C. Dwork, Differential privacy, in: M. Bugliesi, B. Preneel, V. Sassone, I. Wegener (Eds.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1–12.
[61] R. L. Rivest, L. Adleman, M. L. Dertouzos, et al., On data banks and privacy homomorphisms, Foundations of secure computation 4 (11) (1978) 169–180.
[62] J. Konečnỳ, H. B. McMahan, D. Ramage, P. Richtárik, Federated optimization: Distributed machine learning for on-device intelligence, arXiv preprint arXiv:1610.02527.
[63] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, arXiv preprint arXiv:1610.05492.
[64] A. C. Yao, Protocols for secure computations, in: 23rd annual symposium on foundations of computer science (sfcs 1982), IEEE, 1982, pp. 160–164.
[65] M. Sabt, M. Achemlal, A. Bouabdallah, Trusted execution environment: what it is, and what it is not, in: 2015 IEEE Trustcom/BigDataSE/ISPA, Vol. 1, IEEE, 2015, pp. 57–64.
[66] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, arXiv preprint arXiv:1912.04977.
[67] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, Y. Liu, Privacy-preserving blockchain-based federated learning for iot devices, IEEE Internet of Things Journal.
[68] L. Fan, A survey of differentially private generative adversarial networks, in: The AAAI Workshop on Privacy-Preserving Artificial Intelligence, 2020.
[69] I. Mironov, Rényi differential privacy, in: 2017 IEEE 30th Computer Security Foundations Symposium (CSF), IEEE, 2017, pp. 263–275.
[70] T. Van Erven, P. Harremos, Rényi divergence and kullback-leibler divergence, IEEE Transactions on Information Theory 60 (7) (2014) 3797–3820.
[71] J. Dong, A. Roth, W. J. Su, Gaussian differential privacy, arXiv preprint arXiv:1905.02383.
[72] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, J. Sun, Generating multi-label discrete patient records using generative adversarial networks, arXiv preprint arXiv:1703.06490.
[73] U. Asif, B. Mashford, S. Von Cavallar, S. Yohanandan, S. Roy, J. Tang, S. Harrer, Privacy preserving human fall detection using video data, in: Machine Learning for Health Workshop, 2020, pp. 39–51.
[74] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, K. Talwar, Semi-supervised knowledge transfer for deep learning from private training data, arXiv preprint arXiv:1610.05755.
[75] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, Ú. Erlingsson, Scalable private learning with pate, arXiv preprint arXiv:1802.08908.
[76] B. Xin, W. Yang, Y. Geng, S. Chen, S. Wang, L. Huang, Private fl-gan: Differential privacy synthetic data generation based on federated learning, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 2927–2931.
[77] Q. Chang, H. Qu, Y. Zhang, M. Sabuncu, C. Chen, T. Zhang, D. N. Metaxas, Synthetic learning: Learn from distributed asynchronized discriminator gan without sharing medical image data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13856–13866.
[78] N. Phan, Y. Wang, X. Wu, D. Dou, Differential privacy preservation for deep auto-encoders: an application of human behavior prediction., in: Aaai, Vol. 16, 2016, pp. 1309–1316.
[79] X. Zhang, S. Ji, T. Wang, Differentially private releasing via deep generative model (technical report), arXiv preprint arXiv:1801.01594.
[80] L. Xie, K. Lin, S. Wang, F. Wang, J. Zhou, Differentially private generative adversarial network, arXiv preprint arXiv:1802.06739.
[81] A. Triastcyn, B. Faltings, Generating differentially private datasets using gans.
[82] T. Zhang, Z. He, R. B. Lee, Privacy-preserving machine learning through data obfuscation, arXiv preprint arXiv:1807.01860.
[83] G. Acs, L. Melis, C. Castelluccia, E. De Cristofaro, Differentially private mixture of generative neural networks, IEEE Transactions on Knowledge and Data Engineering 31 (6) (2018) 1109–1121.
[84] N. C. Abay, Y. Zhou, M. Kantarcioglu, B. Thuraisingham, L. Sweeney, Privacy preserving synthetic data release using deep learning, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2018, pp. 510–526.
[85] J. Jordon, J. Yoon, M. van der Schaar, Pate-gan: Generating synthetic data with differential privacy guarantees, in: International Conference on Learning Representations, 2018.
[86] Q. Chen, C. Xiang, M. Xue, B. Li, N. Borisov, D. Kaarfar, H. Zhu, Differentially private data generative models, arXiv preprint arXiv:1812.02274.
[87] C. Xu, J. Ren, D. Zhang, Y. Zhang, Z. Qin, K. Ren, Ganobfuscator: Mitigating information leakage under gan via differential privacy, IEEE Transactions on Information Forensics and Security 14 (9) (2019) 2358–2371.
[88] A. Triastcyn, B. Faltings, Generating artificial data for private deep learning, arXiv preprint arXiv:1803.03148.
[89] L. Frigerio, A. S. de Oliveira, L. Gomez, P. Duverger, Differentially private generative adversarial networks for time series, continuous, and discrete open data, in: IFIP International Conference on ICT Systems Security and Privacy Protection, Springer, 2019, pp. 151–164.
[90] T. Takahashi, S. Takagi, H. Ono, T. Komatsu, Differentially private variational autoencoders with term-wise gradient aggregation, arXiv preprint arXiv:2006.11204.
[91] M. Park, J. Foulds, K. Choudhary, M. Welling, Dp-em: Differentially private expectation maximization, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 896–904.
[92] J. M. Abowd, M. J. Schneider, L. Vilhuber, Differential privacy applications to bayesian and linear mixed model estimation, Journal of Privacy and Confidentiality 5 (1).
[93] Y.-X. Wang, J. Lei, S. E. Fienberg, On-average kl-privacy and its equivalence to generalization for max-entropy mechanisms, in: International Conference on Privacy in Statistical Databases, Springer, 2016, pp. 121–134.
[94] G. Hinton, N. Srivastava, K. Swersky, Neural networks for machine learning lecture 6a overview of mini-batch gradient descent, Cited on 14 (8).
[95] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, Image-to-image translation with

conditional adversarial networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[96] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[97] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[98] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in: Advances in neural information processing systems, 2017, pp. 971–980.