Chapter 11
Privacy Guarantees in Synthetic Data

In this chapter, we discuss another important field of applications for synthetic data: ensuring privacy. In many real-world problems, real data is sensitive enough that it is impossible to release. One possible solution could be to train generative models that would produce new synthetic datasets based on real data, while the real data itself would remain secret. But how can we be sure that real data will not be inadvertently leaked? Guarantees in this regard can be provided by the framework of differential privacy. We give a brief introduction to differential privacy, its relation to machine learning, and the guarantees that it can provide for synthetic data generation.

11.1 Why is Privacy Important?

In many domains, real data is not only valuable, but also sensitive; it should be protected by law, commercial interest, and common decency. The unavailability of real data is exactly what makes synthetic data solutions attractive in these domains. But models for generating synthetic data have to train on real datasets anyway, so how do we know we are not revealing it? A number of famous examples show that naive attempts to anonymize data are often insufficient. Let me begin with a few illustrative examples that have become famous in the studies of privacy in computer science.

Probably the first such example dates back to 1997, when the Massachusetts Group Insurance Commission published a carefully anonymized dataset with the medical history of state employees. When it was published, a Ph.D. student from MIT, Latanya Sweeney, spent $20 on a list of all voters from Cambridge, MA (a perfectly legal operation), joined the two datasets according to zip code, birth date, and sex (the three fields that they have in common), and immediately identified William Weld, then the Governor of Massachusetts. She was able to confirm her findings because Weld had a recent public medical incident, but she did not use

S. I. Nikolenko, *Synthetic Data for Deep Learning*, Springer Optimization
and Its Applications 174, https://doi.org/10.1007/978-3-030-75178-4_11

that information in the deanonymization procedure. This was one of the first such incidents that attracted public attention, and Dr. Sweeney's works dating back to the 1990s were among the first not only to raise concerns, but also present specific algorithms with which privacy could be violated even in anonymized data and suggest some ways to efficiently preserve privacy in practice [832, 833].

The second famous example came on August 4, 2006, when *AOL Research*, a division of the internet company AOL that was at the time still a huge internet and email provider (#1 in the United States) and a very popular web portal, published anonymized search queries for over 650,000 users over a 3-month period. Naturally, AOL research did not mean to do any harm: data was released purely for research purposes, and the dataset did not contain anything other than search queries grouped only by user id. But search queries often spoke for themselves: in five days, on August 9, *The New York Times* ran a profile on one of the searchers whom they were able to identify personally from the queries. This profile did not contain anything incriminating, but other search histories were less than innocent: some suggested that the user might be getting ready to commit murder (here's an ethical question for you: should AOL or *Google* be doing anything about this?), and the vagaries of "User 927" even became the basis of an experimental play staged in a Philadelphia theatre in 2008. No, I am not going to cite what this user searched for, but, fortunately and somewhat ironically, this information is just a couple of search queries away...

Search queries are, of course, an easy suspect for revealing sensitive information. *AOL Research* quickly admitted that they made a mistake, removed the dataset in three days (not that it helped, of course), and as a result of the scandal Maureen Govern, the CTO of AOL, resigned in a couple of weeks. But what could go wrong if we publish a much more restricted data type? Say, only the fields where the users do not type in any odd thing? Say, their ratings for products in a recommender engine?..

Alas, the third example is exactly this. Every researcher working in the field of recommender systems knows about the Netflix Prize, a competition held from 2006 to 2009 with a grand prize of one million dollars [64]. This was way before *Kaggle* was a thing, and one could say that *Kaggle* was founded in 2010, in part, as a result of the resounding success of the Netflix Prize. This competition brought to light several new ideas about recommender systems that blossomed into whole directions of research and for a long time defined state of the art in recommender systems [54, 468]. The Netflix Prize dataset only contained the identifiers of movies (names of the movies were known) and ratings that a given user has given them; the users only had numerical ids, there was no personal information disclosed.

However, even this kind of dataset proved to be dangerous: in 2008, researchers from the University of Texas discovered that they could match IMDB user profiles (which are public) with their anonymized Netflix profiles from the published dataset with very high confidence [619]. This means that they could mine information about movie preferences that the users chose not to disclose to their public IMDB profile; needless to say, some of this information might, again, be rather sensitive. As a result, a class action lawsuit was filed against Netflix based on the arguments from [619]; the company settled with the plaintiffs but had to cancel the second Netflix Prize, which had already been announced at the time.

These three examples show that the computational privacy is a very fragile thing. The adversary might have additional information, such as a list of voters or public IMDB profiles. The adversary does not need to attack a large fraction of the dataset because a successful attack on even a small portion of the data might be damaging; after all, the vast majority of people couldn't care less about who knows their movie ratings. A sparse dataset with high-dimensional information about the users helps the adversary: high confidence in the case of the Netflix Prize became possible precisely because there were a lot of movies in the dataset to mine for correlations (about 20,000). And finally, in all three cases, the datasets were not published by malicious hackers or people who didn't know any better: they were published by experienced researchers, in case of AOL and Netflix by researchers who worked in computer science. Still, the adversaries proved to be more resourceful: in these cases, finding a crack in the wall is a much easier job than building a perfect all-encompassing barrier.

When researchers recognized the preservation of privacy as a computer science problem, formal negative results also followed quickly. A famous paper by Dinur and Nissim [199] showed that a few database queries (e.g., taking sums or averages of subsets) suffice to bring about strong violations of privacy even if the database attempts to preserve privacy by introducing noise. Formally, one of their results was that if a database of n private bits $d_1, \ldots, d_n$ responds to queries defined by subsets of bits $S \subseteq \{1, \ldots, n\}$ by specifying the sums $q_S(D) = \sum_{i \in S} d_i$ approximately, and the error in the database's answers is on the order of $o(\sqrt{n})$ (it is hard to imagine a useful database that responds to queries with an error of $\sqrt{n}$ or more!), then a polynomial number of queries suffices to reconstruct *almost all*, i.e., $n - o(n)$ private bits in D. Subsequent results made this even stronger.

This problem also pertains to machine learning. If a machine learning model has trained on a dataset with a few outliers, how do we know it does not "memorize" these outliers directly and will not divulge them to an adversary? For a sufficiently expressive model, such memorization is quite possible, and note that the outliers are usually the most sensitive data points. For instance, Carlini et al. [114] show that state of the art language models do memorize specific sequences of symbols, and one can extract, e.g., a secret string of numbers from the original dataset with a reasonably high success rate.

How does all this relate to synthetic data? Machine learning on privacy-sensitive datasets might be an important field of application for synthetic data: wouldn't it be great if AOL or Netflix didn't have to publish their *real* datasets but would publish information about *synthetic* users instead? This would immediately alleviate all privacy concerns. On the other hand, choosing a uniform distribution for the ratings or random character strings for search queries would render such synthetic datasets completely useless: naturally, to be useful the distribution of synthetic data must resemble the distribution of real data. But then wouldn't we be divulging private information? Looks like we need to dig a little deeper.

11.2 Introduction to Differential Privacy

The field of *differential privacy*, pioneered by Dwork et al. [214, 216, 218], was largely motivated by considerations such as the ones we saw in the previous section. The works of Cynthia Dwork have been widely recognized as some of the most novel and interesting advances in modern computer science: Prof. Dwork received the Dijkstra Prize in 2007, the Gödel Prize in 2017 for the work [216], the Hamming Medal and the Knuth Prize in 2020.

In the main definition of the field, a mechanism (randomized algorithm) M is called (ϵ, δ)-*differentially private* for some positive real parameters ϵ and δ if for any two databases D and D' that differ in only a single point x, $D \setminus \{x\} = D' \setminus \{x\}$, and any subset of outputs S

$$p\left(M(D) \in S\right) \leq e^{\epsilon} p\left(M(D') \in S\right) + \delta,$$

or, equivalently, for every point s in the output range of M

$$\left| \ln \frac{p\left(M(D) = s\right)}{p\left(M(D') = s\right)} \right| \leq \epsilon \quad \text{with probability } 1 - \delta.$$

The ratio $\ln \frac{p(M(D)=s)}{p(M(D')=s)}$ is an important quantity called *privacy loss* that needs to be bounded in absolute value.

The intuition here is that an adversary who receives only the outputs of M should have a hard time learning anything about any single point in D. The same intuition could be reformulated in terms of a Bayesian update of beliefs (recall Section 2.2, where we discussed how the Bayes theorem is the foundation of all machine learning): an adversary, after learning the result $M(D) = s$, updates their beliefs about the two databases (that is, about the question whether the dataset contains some specific point x) as

$$\frac{p(D \mid M(D) = s)}{p(D' \mid M(D) = s)} = \frac{p(D)}{p(D')} \frac{p(M(D) = s \mid D)}{p(M(D') = s \mid D')},$$

and the latter ratio on the right-hand side is precisely the privacy loss whose logarithm's absolute value is bounded by ϵ in the definition.

This definition has a number of important desirable qualities. First, it is robust to the introduction of *additional information*, that is, knowledge of some events available to an adversary: naturally, additional information regarding the database will help the adversary, but the definition still remains in place: an (ϵ, δ)-differentially private mechanism will remain (ϵ, δ)-differentially private and will not help the adversary further. Second, it is immune to *postprocessing*: an adversary cannot compute some function of the private mechanism's result $M(D)$ and compromise the privacy, i.e., the privacy loss cannot be increased by thinking hard about the results of M. Third, it is *composable*: if an adversary has access to two mechanisms, M_1 which is (ϵ_1, δ_1)-differentially private and M_2 with parameters (ϵ_2, δ_2), any composition of them will have parameters not exceeding $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ regardless

of whether M_1 and M_2 know about each other; this allows for modular design of private architectures. Fourth, it allows for *group privacy*, that is, when the databases differ by k elements an $(\epsilon, 0)$-differentially private mechanism will become at most $(k\epsilon, 0)$-differentially private.

Unfortunately, this definition conceals an unpleasant tradeoff. If we set $\delta = 0$ the definition becomes too strong: for example, it is too pessimistic for repeated applications of M (the exponent grows linearly). But if not, δ may hide a complete failure of privacy preservation: for instance, an (ϵ, δ)-differentially private mechanism may reveal the entire database with probability δ or reveal the δ share of data with probability 1. Therefore, in practice, it should hold that $\delta \ll \frac{1}{n}$.

I do not want to get to a much deeper discussion of differential privacy than the definitions, so I will conclude this brief intro with an example of the *Laplace mechanism*, probably the simplest and most classical example of a differentially private mechanism. Suppose that we are sending numerical queries to a database of n integer numbers, that is, a query is a function $f : \mathbb{N}^n \to \mathbb{R}^k$. An important property of such functions f in this case is their L_1-*sensitivity*, a measure of how much changing a single element in the database can change the function:

$$\Delta f = \max_{D, D' : D \text{ and } D' \text{ differ in one point}} \left\| f(D) - f(D') \right\|_1 .$$

The Laplace mechanism works as follows: when someone asks to compute $f(D)$, it computes the correct answer and gives out a version of it perturbed by the Laplace distribution (hence the name):

$$M_L(D, f, \epsilon) = f(D) + \left(Y_1\ Y_2\ \ldots\ Y_k\right), \text{ where } Y_i \sim \text{Lap}\left(\frac{\Delta f}{\epsilon}\right)$$

for some constant ϵ and for the Laplace distribution

$$\text{Lap}(x \mid b) = \frac{1}{2b} e^{-\frac{1}{b}|x|}.$$

Let us now compare the distributions of M_L results on two databases D and D' that differ at a single point. For some point $\mathbf{z} \in \mathbb{R}^k$,

$$\frac{p(M_L(D, f, \epsilon) = \mathbf{z})}{p(M_L(D', f, \epsilon) = \mathbf{z})} = \prod_{i=1}^{k} \frac{e^{-\frac{\epsilon}{\Delta f}|f(D)_i - z_i|}}{e^{-\frac{\epsilon}{\Delta f}|f(D')_i - z_i|}} =$$

$$= \prod_{i=1}^{k} e^{\frac{\epsilon(|f(D')_i - z_i| - |f(D)_i - z_i|)}{\Delta f}} \le \prod_{i=1}^{k} e^{\frac{\epsilon(|f(D')_i - f(D)_i|)}{\Delta f}} = e^{\frac{\epsilon \|f(D) - f(D')\|_1}{\Delta f}} \le e^{\epsilon},$$

where the first inequality is the triangle inequality and the second is by our assumption on the L_1-sensitivity of f. Similarly, $\frac{p(M_L(D, f, \epsilon) = \mathbf{z})}{p(M_L(D', f, \epsilon) = \mathbf{z})} \ge e^{-\epsilon}$, and we have proved that the Laplace mechanism is $(\epsilon, 0)$-differentially private.

A similar (but much more involved and cumbersome) argument shows that the same can be achieved with L_2-sensitivity and Gaussian noise. In other words, if we define L_2-sensitivity as

$$\Delta_2 f = \max_{D,D' : D \text{ and } D' \text{ differ in one point}} \left\| f(D) - f(D') \right\|_2$$

and define the Gaussian mechanism as

$$M_{\mathcal{N}}(D, f, \epsilon) = f(D) + \left(Y_1\ Y_2\ \ldots\ Y_k\right), \text{ where } Y_i \sim \mathcal{N}\left(0, \sigma^2\right),$$

then $M_{\mathcal{N}}$ will be (ϵ, δ)-differentially private if we let

$$\sigma \geq c\frac{\Delta_2 f}{\epsilon}, \quad \text{where} \quad c^2 > 2\ln\frac{1.25}{\delta};$$

see, e.g., [218] for details.

11.3 Differential Privacy in Deep Learning

We are most interested in machine learning applications for privacy: how can we give access to the results of learning without giving access to the training data? Before we proceed to applying differential privacy to machine learning, we should define a formal setting for such considerations. What should we allow the adversary to do? One might think that we can hide the model from the adversary, but both theory and practice show that if we provide an interface for running inference on the model (which we definitely have to assume), a smart adversary can learn so much about the model that it doesn't make much sense to distinguish these two cases. Therefore, research in this field mostly concentrates on how to keep training data private while giving the model and its weights to the adversary (the "white box" scenario).

Note that a model that generalizes well does not necessarily preserve privacy. Generalization is an average-case notion, and it characterizes how well the model's accuracy (or another objective function) transfers to new data. Privacy, on the other hand, is a worst-case notion, and it deals with the corner cases and the information that the entire model provides, not just its performance. For example, if you train an SVM for classification, it might generalize very well, but the model will explicitly contain (and thus provide to any adversary) full and unperturbed information about its support vectors, which can hardly be called privacy-preserving. And let's not even get started on nearest neighbors...

Using a "standard model" that has been tried and tested also doesn't really help. For example, Zhang et al. [992], in a very important paper that has already become a classic of deep learning research, studied standard models such as *AlexNet* on standard datasets such as *ImageNet* (we discussed *AlexNet* in Section 3.2). Their

experiment was to introduce a *random permutation* of the labels, that is, assign labels from 1000 *ImageNet* classes completely at random, thus making the dataset entirely unlearnable. After training *AlexNet*, they indeed saw purely random-looking accuracy on the test set (about 0.1% top-1 accuracy and about 0.5% top-5 accuracy), but on the training set the model actually achieved more than 90% top-1 accuracy, not much worse than after training on original labels with the exact same learning parameters! This means that even in the absence of any possibility for generalization and extracting useful features, modern deep learning models can learn quite a lot by simply memorizing the data; note also that *AlexNet* is by modern standards a pretty small and weak network...

Therefore, if we want to be able to train on real data, we need to somehow introduce privacy-preserving transformations into the model training process. Since, in this book, we are mainly interested in deep learning, I will not go into preserving privacy with other machine learning models; there is a growing body of research in this field, and I can refer, e.g., to the surveys [238, 287, 312, 400] and references therein. Our focus in this section is on how to make complex high-dimensional optimization, such as training deep neural networks with stochastic gradient descent (recall Sections 2.4 and 2.5), respect privacy constraints. As we have already seen, the basic approach to achieving differential privacy is to add noise to the output of M, just like the Laplace and Gaussian mechanisms do. Many classical works on the subject focus on estimating and reducing the amount of noise necessary to ensure privacy under various assumptions [214–217, 520]. However, it is not immediately obvious how to apply this idea to a deep neural network. There are two major approaches to achieving differential privacy in deep learning, that is, in stochastic gradient descent.

The most important advance in this field came from Abadi et al. [1], who suggested a method for controlling the influence of the training data during stochastic gradient descent called *Differentially Private SGD* (DP-SGD). They use the Gaussian mechanism that we introduced in the previous section, so the basic idea is to add Gaussian noise to the gradients on every step of the SGD. But in order to estimate the variance σ for the necessary noise, the Gaussian mechanism needs to know an estimate on the influence that each individual example can have on the gradient $\mathbf{g}_k$ computed on the minibatch at step k. How can we get such an estimate when we do not have any prior bound on the gradients? We have to bound them ourselves!

Specifically, Abadi et al. clip the gradients on each SGD iteration to a predefined value of the L_2-norm and add Gaussian noise to the resulting gradient value. The entire DP-SGD scheme is presented in Algorithm 9. By careful analysis of the privacy loss variable, i.e., $\log \frac{p(A(D)=s)}{p(A(D')=s)}$ above, Abadi et al. show that the resulting algorithm preserves differential privacy under reasonable choices of the clipping and random noise parameters. Moreover, this is a general approach that is agnostic to the network architecture and can be extended to various first-order optimization algorithms based on SGD.

A year later, Papernot et al. [653] (actually, mostly the same group of researchers from *Google*) presented the *Private Aggregation of Teacher Ensembles* (PATE) approach. In PATE, the final "student" model is trained from an ensemble of "teacher" models that have access to sensitive data, while the "student" model only has access

Algorithm 9: Differentially private stochastic gradient descent

Initialize $\mathbf{w}_0$, $k := 0$;
repeat
  $D_k := \text{Sample}(D)$;
  for $d \in D_k$ **do**
    $\mathbf{g}_k(d) := \nabla_{\mathbf{w}} f\ (\mathbf{w}_k, d)$;
    $\bar{\mathbf{g}}_k(d) := \mathbf{g}_k(d) / \max\left(1, \frac{1}{C} \|\mathbf{g}_k(d)\|_2\right)$;
  end
  $\mathbf{g}_k := \frac{1}{|D_k|}\left(\sum_{d \in D_k} \bar{\mathbf{g}}_k(d) + \mathcal{N}\left(0, \sigma^2 C^2 \mathbf{I}\right)\right)$;
  $\mathbf{w}_{k+1} := \mathbf{w}_k - \alpha_k \mathbf{g}_k$;
  $k := k + 1$;
until *a stopping condition is met*;

to (noisy) aggregated results of "teacher" models, which allows to control disclosure and preserve privacy. A big advantage of this approach is that "teacher" models can be treated as black box while still providing rigorous differential privacy guarantees based on the same moments accounting technique from [1]. Incidentally, the best results were obtained with adversarial training for the "student" in a semi-supervised fashion, where the entire dataset is available for the "student" but labels are only provided for a subset of it, preserving privacy.

In conclusion, I think it is important to note the practical side of things. Differential privacy is a worst-case theoretical concept, and definitions of an (ϵ, δ)-differentially private mechanism might have reminded the reader of definitions from theoretical cryptography, where usually nothing is possible to actually achieve and even the best results are often either negative or impossible to apply in practice. But differential privacy for deep learning is a field that has actual implementations. The original paper by Abadi et al. was already accompanied by a repository that added differentially private variations of *Tensorflow* optimizers[1]. And the latest news is the release of *Opacus*, a library developed by *Facebook* researchers Davide Testuggine and Ilya Mironov that enables differential privacy for *PyTorch* models[2].

Thus, deep learning with differential privacy guarantees may eventually provide a good answer to the problem of preserving information regarding the datasets. But if you want to release a *dataset* for the general public, say organize a *Kaggle* competition, rather than just publish your model while keeping the original dataset private, you still cannot avoid the generation of synthetic data with privacy guarantees. This is exactly what we will discuss in the next section.

[1] At the time of writing (late 2020), the *Tensorflow Privacy* library is alive and well supported: https://github.com/tensorflow/privacy.

[2] At the time of writing (late 2020), this library has been very recently released, so it obviously also does not lack support: https://github.com/pytorch/opacus

11.4 Differential Privacy Guarantees for Synthetic Data Generation

In this section, we review the applications of differential privacy and related concepts to synthetic data generation. The purpose is similar: the release of a synthetic dataset generated by some model trained on real data should not disclose information regarding the individual points in this real dataset. Our review is slanted towards deep learning; for a more complete picture of the field, we refer to the surveys in [71, 214]. However, we do note the efforts devoted to generating differentially private synthetic datasets in classical machine learning. In particular, Lu et al. [559] develop a model for making sensitive databases private by fixing a set of queries to the database and perturbing the outputs to ensure differential privacy. Zhang et al. present the *PrivBayes* approach [997]: construct a Bayesian network that captures the correlations and dependencies between data attributes, inject noise into the marginals that constitute this network, and then sample from the perturbed network to produce the private synthetic dataset. In a similar effort, the *DataSynthetizer* model by Ping et al. [673] is able to take a sensitive dataset as input and generate a synthetic dataset that has the same statistics and structure but at the same time provides differential privacy guarantees.

We also note some privacy-related applications of synthetic data that are not about differential privacy. For example, Ren et al. [721] present an adversarial architecture for video face anonymization; their model learns to modify the original real video to remove private information while at the same time still maximizing the performance of action recognition models (see also Section 6.6).

The general approaches we have discussed in the previous section have been modified and applied for producing synthetic data with generative models, mostly, of course, with generative adversarial networks. Although the methods are similar, we note an important conceptual difference that synthetic data brings in this case. *Model release* approaches in the previous section assumed access to and full control of model training. *Data release* approaches (here we use the terminology from [871]) that perform synthetic data generation have the following advantages:

- they can provide private data to third parties to construct better models and develop new techniques or use computational resources that might be unavailable to the holders of sensitive data;
- moreover, these third parties are able to pool synthetic data from different sources, while in the model release framework this would require a transfer of sensitive data;
- synthetic data can be either traded of freely made public, which is an important step towards reproducibility of research, especially in such fields as bioinformatics and healthcare, where reproducibility is an, especially, important problem and where, at the same time, sensitive data abounds.

In this section, we discuss existing constructions of GANs that provide rigorous privacy guarantees for the resulting generated data. Basically, in the ideal case, a

differentially private GAN has to generate an artificial dataset that would be sampled from the same distribution p_{data} but with differential privacy guarantees as discussed above. One general remark that is used in most of these works is that in a GAN-based architecture, it suffices to have privacy guarantees or additional privacy-preserving modifications (such as adding noise) only in the discriminator since gradient updates for the generator are functions of discriminator updates. Another important remark is that in cases when we generate differentially private synthetic data, a drop in quality for subsequent "student" models trained on synthetic data is expected in nearly all cases, not because of any deficiencies of synthetic data vs. real in general but because the nature of differential privacy requires adding random noise to the generative model training.

Xie et al. [955] present the differentially private GAN (DPGAN) model, which is basically the already classical Wasserstein GAN [27, 303] but with additional noise on the gradient of the Wasserstein distance, in a fashion following the DP-SGD approach (Section 11.3). They apply DPGAN to generate electronic health records, showing that classifiers trained on synthetic records have accuracy approaching that of classifiers trained on real data, while guaranteeing differential privacy. This was further developed by Zhang et al. [1002], who used the Improved WGAN framework [303] and obtained excellent results on the synthetic data generated from various subsets of the LSUN dataset [1002], which is already a full-scale image dataset, albeit at low resolution (64×64).

Beaulieu-Jones et al. [50] apply the same idea to generating electronic health records, specifically training on the data of the Systolic Blood Pressure Trial (SPRINT) data analysis challenge [205, 854], which are in nature low-dimensional time series. They used the DP-SGD approach for the Auxiliary Classifier GAN (AC-GAN) architecture [637] and studied how the accuracy of various classifiers drops when passing to synthetic data. Triastcyn and Faltings [871] continue this line of work and show that differential privacy guarantees can be obtained by adding a special Gaussian noise layer to the discriminator network. They show good results for "student" models trained on synthetically generated data for MNIST, but already at the SVHN dataset the performance degrades more severely.

Bayesian methods are a natural fit for differential privacy since they deal with entire distributions of parameters and lend themselves easily to adding extra noise needed for DP guarantees. In a combination of generative models and Bayesian methods, a Bayesian variant of the GAN framework, which provides representations of full posterior distributions over the parameters, was provided by Saatchi and Wilson [747]. The idea of their *Bayesian GAN* is to introduce prior distributions on generator parameters $\boldsymbol{\theta}_g$ and discriminator parameters $\boldsymbol{\theta}_d$, $p\left(\boldsymbol{\theta}_g \mid \boldsymbol{\alpha}_g\right)$, and $p\left(\boldsymbol{\theta}_d \mid \boldsymbol{\alpha}_d\right)$, respectively, and infer posteriors over $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_d$

$$p\left(\boldsymbol{\theta}_g \mid Z, \boldsymbol{\theta}_d\right) \propto p\left(\boldsymbol{\theta}_g \mid \boldsymbol{\alpha}_g\right) \prod_{n=1}^{N_g} D\left(G\left(\mathbf{z}_n; \boldsymbol{\theta}_g\right); \boldsymbol{\theta}_d\right),$$

$$p\left(\boldsymbol{\theta}_d \mid Z, X, \boldsymbol{\theta}_g\right) \propto p\left(\boldsymbol{\theta}_d \mid \boldsymbol{\alpha}_d\right) \prod_{n=1}^{N_d} D\left(\mathbf{x}_n; \boldsymbol{\theta}_d\right) \prod_{n=1}^{N_g} \left(1 - D\left(G\left(\mathbf{z}_n; \boldsymbol{\theta}_g\right); \boldsymbol{\theta}_d\right)\right),$$

where $\mathbf{x}_n$ are real inputs, $\mathbf{z}_n$ are random noise samples, and N_d and N_g are the numbers of real and fake samples, respectively.

Saatchi and Wilson provide learning algorithms in this setting, marginalizing the above posteriors over random noise by Monte Carlo integration and sampling from posterior distributions with stochastic gradient Hamiltonian Monte Carlo [136, 1033]. Arnold et al. [30] adapted the BayesGAN framework for differential privacy by injecting noise into the gradients during training, which was shown by Wang et al. [918] to lead to DP guarantees. They apply the resulting *DP-BayesGAN* framework to microdata, i.e., medium-dimensional samples of 40 explanatory variables of different nature and one dependent variable.

As for the PATE framework, it cannot be directly applied to GANs since noisy aggregation of a PATE ensemble is not a differentiable function that could serve as part of a GAN discriminator. Ács et al. [6] proposed to use a differentially private clustering method to split the data into k clusters, then train a separate generative models (the authors tried VAE) on their own clusters, and then create a mixture of the resulting models that would inherit differential privacy properties as well. A recent work by Jordon et al. [976] circumvents the non-differentiability problem by training a "student-discriminator" on already differentially private synthetic data produced by the generator. The learning procedure alternates between updating "teacher" classifiers for a fixed generator on real samples and updating the "student-discriminator" classifier and the generator for fixed "teachers". PATE-GAN works well on low-dimensional data but begins to lose ground on high-dimensional datasets such as, e.g., the UCI Epileptic Seizure Recognition dataset (with 184 features).

However, these results are still underwhelming; it has proven very difficult to stabilize GAN training with the additional noise necessary for differential privacy guarantees, which has not allowed researchers to progress to, say, higher resolution images so far. In a later work, Triastcyn and Faltings [870] consider a different approach: they use the *empirical DP* framework [3, 124, 168, 768], an approach that empirically estimates the privacy of a posterior distribution, and the modification that ensures privacy is usually a sufficiently diffuse prior. In this framework, evaluating the privacy would reduce to training a GAN on the original dataset D, removing one sample from D to obtain D', retraining the GAN and comparing the probabilities of all outcomes, and so on, repeating these experiments enough times to obtain empirical estimates for ϵ and δ. For realistic GANs, a large number of retrainings is impractical, so Triastcyn and Faltings modify this procedure to make it operate directly on the generated set $\tilde{D}$ rather than the original dataset D. They study the tradeoff of privacy vs. accuracy of the "student" models trained on synthetic data

and show that GANs can fall into the region of practical values for both privacy and accuracy. Their proposed modification of the architecture (a single randomizing layer close to the end of the discriminator) strengthens DP guarantees while preserving good generation quality for datasets up to *CelebA* [542]; in fact, it appears to serve as a regularizer and improve generation.

Frigerio et al. [246] extend the DPGAN framework to continuous, categorical, and time series data. They use the Wasserstein GAN loss function [303], extending the moment accountant to this case. To handle discrete variables, the generator produces an output for every possible value with a softmax layer on top, and its results are sent to the discriminator. Bindschadler [71] presents a *seedbased* modification of synthetic data generation: an algorithm that produces data records through a generative model conditioned on some seed real data record; this significantly improves quality but introduces correlations between real and synthetic data. To avoid correlations, Bindschadler introduces privacy tests that reject unsuitable synthetic data points. The approach can be used in complex models based on encoder-decoder architecture by adding noise to a seed in the latent space; it has been evaluated across different domains from census data to celebrity face images, the latter through a VAE/GAN architecture [497].

Finally, we note that synthetic data produced with differential privacy guarantees is also starting to gain legal status; in a technical report [58], Bellovin et al. from the Stanford Law School discuss various definitions of privacy from the point of view of what kind of data can be released. They conclude: "...as we recommend, synthetic data may be combined with differential privacy to achieve a best-of-both-worlds scenario", i.e., combining added utility of synthetic data produced by generative models with formal privacy guarantees.

11.5 Case Study: Synthetic Data in Economics, Healthcare, and Social Sciences

Synthetic data is increasingly finding its way into economics, healthcare, and social sciences in a variety of applications. We discuss this set of models and applications here since often the main concern that drives researchers in these fields to synthetic data is not lack of data *per se* but rather privacy issues. A number of models that guarantee differential privacy have already been discussed above, so in this section, we concentrate on other approaches and applications.

As long ago as 1993, Rubin [740] discussed the dangers of releasing microdata (i.e., information about individual transactions) and the extremely complicated legal status of data releases, as the released data might be used to derive protected information even if it had been masked by standard techniques. To avoid these complications, Rubin proposed to use imputed synthetic data instead: given a dataset with confidential information, "forget" and impute confidential values for a sample from this dataset, using the same background variables but drawing confidential

data from the predictions of some kind of imputation model. Repeating the process for several samples, we get a multiply-imputed population that can then be released. In the same year (actually, the same special issue of the *Journal of Official Statistics*), Little [530] suggested to also keep the non-confidential part of the information to improve imputation. By now, synthetic datasets produced by multiple imputation are a well-established field of statistics, with applications to finance and economics [195, 713], healthcare [18], social sciences [102], survey statistics [4, 13], and other domains. Since the main emphasis of the present survey is on synthetic data for deep learning, we do not go into details about multiple imputation and refer to the book [207] and the main recent sources in the field [206, 700, 712, 714].

In a very recent work, Heaton and Witte [332] propose another interesting take on synthetic data in finance. They begin with the well-known problem of overfitting during backtesting: since there is a very large number of financial products and relatively short time series available for them, one can always find a portfolio (subset of products) that works great during backtesting, but it does not necessarily reflect future performance. The authors suggest to use synthetic data not to train financial strategies (they regard it as infeasible), but rather to *evaluate* developed strategies, generating synthetic data with a different distribution of abnormalities and testing strategies for robustness in these altered circumstances. Interestingly, the motivation here is not to improve or choose the best strategies, but to obtain evidence of their robustness that could be used for regulatory purposes. As a specific application, the authors use existing fraud detection algorithms to find anomalies in the Kaggle Credit Card Fraud Detection Dataset [180] and generate synthetic data that balances the found abnormalities.

However, at present, I know of no direct applications where synthetically generated financial time series that would lead to improved results in financial forecasting, developing financial strategies, and the like. In general, financial time series are notorious for not being amenable to either prediction or accurate modeling, and even with current state-of-the-art economic models, it looks like generating useful synthetic financial time series is still in the future. Moreover, the reasons we discussed in Section 8.4 regarding why synthetic data is unpopular in natural language processing apply here as well.

As for healthcare, this is again a field where the need for synthetic data was understood very early, and this need was mostly caused by privacy concerns: hospitals are required to protect the confidentiality of their patients. Ever since the first works in this direction, dating back to early 1990s, researchers mostly concentrated on generating synthetic electronic medical records (EMR) in order to preserve privacy [44]. Among more recent works, MDClone [594] is a system that samples synthetic EMRs from the distributions learned on existing cohorts, without actually reusing original data points. Walonoski et al. [898] present the *Synthea* software suite designed to simulate the lifespans of synthetic patients and produce realistic synthetic EMRs. McLaghlan [591] discusses realism in synthetic EMR generation and methods for its validation, and in another work presents a state transition machine that incorporates domain knowledge and clinical practice guidelines to generate realistic synthetic EMRs [592].

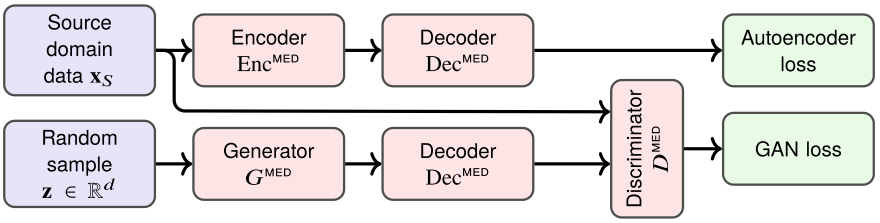

Fig. 11.1 The architecture of *medGAN* [150]. Blocks with identical labels have shared weights.

Another related direction of research concentrates not on individual EMRs, but on modeling entire populations of potential patients. Synthetic micro-populations produced by Smith et al. [805] are intended to match various sociodemographic conditions found in real cities and use them in imitational modeling to estimate the effect of interventions. Moniz et al. [610, 746] create synthetic EMRs made available on the CDC Public Health grid for imitational modeling. Buczak et al. [98] generate synthetic EMRs for an outbreak of a certain disease (together with background records). Kartoun [441] progressed from individual EMRs to entire virtual patient repositories, concentrating on preserving the correct general statistics while using simulated individual records. However, most of this work does not make use of modern formalizations of differential privacy or recent developments in generative models, and only very recently researchers have attempted to bring those into the healthcare microdata domain as well.

A direct application of GANs for synthetic EMR generation was presented by Choi et al. [150]. Their *medGAN* model consists of a generator G^{MED}, discriminator D^{MED}, and an autoencoder with encoder Enc^{MED} and decoder Dec^{MED}; the architecture is shown in Fig. 11.1. The autoencoder is trained to reconstruct real data $\mathbf{x}_T \sim \mathcal{X}_T$, while G^{MED} learns to generate latent representations $G^{\text{MED}}(\mathbf{z})$ from a random seed $\mathbf{z}$ such that D^{MED} will not be able to differentiate between $\text{Dec}^{\text{MED}}(G^{\text{MED}}(\mathbf{z}))$ and a real sample $\mathbf{x}_T \sim \mathcal{X}_T$. Privacy in the *medGAN* model is established empirically, and the main justification for privacy is the fact that *medGAN* uses real data only for the discriminator and never trains the generator on any real samples. We note, however, that in terms of generation *medGAN* is not perfect: for example, Patel et al. [659] present the *CorrGAN* model for correlated discrete data generation (with no regard for privacy) and show improvements over *medGAN* with a relatively straightforward architecture.

The DP-SGD framework has also been applied to GANs in the context of medical data. We have discussed Beaulieu-Jones et al. [50] above. Another important application for synthetic data across many domains, including but not limited to finance, would be to generate synthetic time series. This, however, has proven to be a more difficult problem, and solutions are only starting to appear. In particular, Hyland et al. [228] present the Recurrent GAN (RGAN) and Recurrent Conditional GAN architectures designed to generate realistic real-valued multi-dimensional time series. They applied the architecture to generating medical time series (vitals measured for

ICU patients) and reported successful generation and ability to train classifiers on synthetic data, although there was a significant drop in quality when testing on real data. Hyland et al. also discuss the possibility to use a differentially private training procedure, applying the DP-SGD framework to the discriminator and thus achieving differential privacy for the RGAN training. The authors report that after this procedure, synthetic-to-real test results deteriorate significantly but remain reasonable in classification tasks on ICU patient vitals.

Finally, we note another emerging field of research related to generating synthetic EMRs for the sake of privacy: generating clinical notes and free-text fields in EMRs with neural language models (see also Section 8.4). Latest advances in deep learning for natural language processing have led to breakthroughs in large-scale language modeling [193, 359, 697], and it has been applied to smaller datasets of clinical notes as well. Lee [505] uses an encoder-decoder architecture to generate chief complaint texts for EMRs. Guan et al. [302] propose a GAN architecture called *mtGAN* (medical text GAN) for the generation of synthetic EMR text. It is based on the SeqGAN architecture [982] and is trained with the REINFORCE algorithm; the primary difference is a condition added by Guan et al. to be able to generate EMRs for a specific disease or other features. Melamud and Shivade [597] compare LSTM-based language models for generating synthetic clinical notes, suggesting a new privacy measure and showing promising results. Further advances in this direction may be related to the recently developed differentially private language models [593].

11.6 Conclusion

In this chapter, we have investigated a motivation for synthetic data very different from the rest of the book. Instead of making synthetic data that alleviates the hardships of collecting and labeling real data, here we have been making synthetic data because real data, while available, cannot be published for legal or ethical reasons. This motivation has led to very different methods: now the main problem is to guarantee that real data is not released by publishing the synthetic dataset. Therefore, in this chapter, we have been mostly talking about the main (and probably the only) approach that can provide these guarantees: differential privacy.

Next, we come to the final chapter of the book. In it, we will try to highlight the most promising directions for further research, ideas that I believe will bring interesting advances in the nearest future. Who knows, maybe my readers will be among those who take up these directions and bring machine learning to new heights with synthetic data. Let's find out together!