Microsoft

Microsoft SmartNoise Differential Privacy Machine Learning Case Studies

By Andreas Kopp

Digital Advisor for AI solutions

Special thanks to Joshua Allen, Andreas Becker, Sarah Bird, Ethan Cowan, Eduardo de Leon, Simone Droll, Anja Fiegler, Jörg Finkeisen, Julia Kabalar, Thomas Langkabel, Alexander Loth, Lucas Rosenblatt, Habiba Sarhan, Michael Shoemate, Timm Walz, and Kevin White.

Contents

Interactive Demo Case Studies 1
Introduction 2
Differential Privacy Overview 3
The Dilemma of Traditional Data Anonymization Practices 3
The Differential Privacy Concept 6
Differential Privacy Applications 4
The SmartNoise System 7
Protecting against Privacy Attacks 8
The Impact of Privacy and the Amount of Data on Statistical Accuracy 14
Differentially Private Machine Learning 17
Differentially Private Machine Learning Algorithms 19
Machine Learning Based on a Differentially Private Dataset 25
Deep Learning on Medical Images 28
Conclusion 40
References 41

Interactive Demo Case Studies

We have developed the following Jupyter notebooks so that you can experience the concepts discussed in this paper in practice and adapt them to your own use cases.

Protecting Statistics Against Reconstruction Attacks
Learn how attackers might reconstruct sensitive income information based on released summary statistics. SmartNoise can help you protect personal data against reconstruction attacks.

Protecting Sensitive Data Against Re-Identification Attacks
Learn how attackers might combine an anonymized medical dataset with other available data to identify patients. See how to use SmartNoise to protect personal data against re-identification attacks.

Privacy-Preserving Statistical Analysis
Learn how to use SmartNoise to disclose statistical reports with the Differential Privacy concept.
Understand how different levels of privacy guarantees and data set sizes impact statistical accuracy.

Machine Learning Using a Differentially Private Classifier
Check out different options to perform differentially private machine learning for a classification task.
Experience how different levels of privacy guarantees and data set sizes affect model quality.

Generating a Synthetic Dataset for Privacy-Preserving Machine Learning
See how SmartNoise can be used to generate a differentially private dataset that can be disclosed without privacy concerns. Check out how the synthetic dataset can be used for machine learning.

Detect Pneumonia in X-Ray Images while Protecting Patients' Privacy
Discover how to perform differentially private deep learning by analyzing medical images.

Introduction

Sensitive and confidential information about individuals is extensively used and shared between companies, government entities, research organizations, and other parties. Inadequate usage of this kind of information can result in significant consequences, such as harm to an individual's reputation, employability, creditworthiness, and insurability.

Pioneered by Microsoft Research and their collaborators, Differential Privacy is the gold standard for protecting individuals' data in applications like preparing and publishing statistical analyses. Differential Privacy provides a mathematically measurable privacy guarantee to individual data subjects. It offers significantly higher privacy levels than commonly used disclosure limitation practices like data anonymization. The latter increasingly shows vulnerability to re-identification attacks – especially as more data about individuals become publicly available.

This whitepaper provides practical guidance on how personal data can be rigorously protected for applications like statistics, machine learning, and deep learning using Differential Privacy. To make the beneficial and fascinating concept of Differential Privacy accessible to a broad audience, we refrain from discussing the underlying mathematical concepts. Rather, we seek to keep the technical descriptions at a high level. Nonetheless, we recommend that readers have background knowledge about and understand machine learning concepts.

Decisions based on artificial intelligence algorithms increasingly impact our lives. There are many open questions left for the ongoing debate about ethical challenges. In collaboration with policy makers, research institutions, and other technology companies, Microsoft is working on principles, best practices, and tools for a responsible approach to creating and using artificial intelligence solutions. The following diagram provides an overview of Microsoft's Responsible AI framework.

Figure 1: Microsoft Responsible AI Principles

Differential Privacy is a vital concept for protecting personal data and therefore provides one option for covering the privacy part in the "Privacy & Security" section. That said, it is not designed to increase the security of systems used to develop or operate machine learning models. Other approaches can be leveraged to accomplish this task (e.g., Confidential Machine Learning).

Differential Privacy Overview

The Dilemma of Traditional Data Anonymization Practices

Many people would intuitively argue as follows: "*If we remove the personally identifiable part from the records of our datasets before release, then privacy is ensured*." Standard practices support this *false conclusion* since data anonymization techniques are prevalent.

Let's look at hospital representatives, who aim to release medical treatment insights for research purposes to improve future therapies. They have several options to choose the data categories and the level of detail of the disclosed data, as shown in the illustration below. The privacy protection objective is to prevent any conclusions about individuals might be drawn from the published data.

Name	Gender	Age	Zip	Diagnosis	Treatment Code	Outcome
Michael Hampton	M	22	21221	MS	025	0
Erik Chung	M	34	23411	Pneumonia	083	2
Jacqueline Peters	F	37	46378	Cancer	044, 045	1
Derek Wiggins	M	45	51554	Pneumonia	087	2
Holly Flores	F	77	26387	Pneumonia	083	1
Chelsea Riley	F	30	43211	Cancer	045	0
Michael Roy	M	82	02742	Diabetes	033	2
Mary Simon	F	28	89011	MS	025	1
Stephen Turner	M	40	51729	Pneumonia	083	2
...	...	...	...	...	...	...

Original dataset

1 Reducing personally identifiable information

2 Providing aggregate statistics

Gender	Age	Zip	Diagnosis	Treatment Code	Outcome
M	20-29	21***	MS	025	0
M	30-39	23***	Pneumonia	083	2
F	30-39	46***	Cancer	044, 045	1
M	40-49	51***	Pneumonia	087	2
F	70-79	26***	Pneumonia	083	1
F	30-39	43***	Cancer	045	0
M	80+	02***	Diabetes	033	2
F	20-29	89***	MS	025	1
M	40-49	51***	Pneumonia	083	2
...	...	...	...	...	...

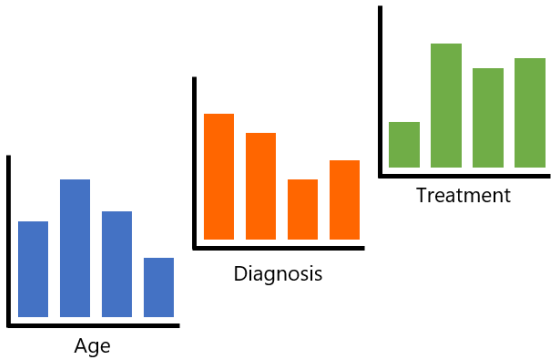


Figure 2: Data Anonymization Examples

The crucial problem with anonymized data is that the released records often include unique combinations of variables (digital fingerprints) that someone might link to other publicly available information to re-identify specific people. For instance, research has shown that 87% of Americans can be uniquely identified with only three pieces of data: Gender, birthday, and ZIP code.[1]

Today's data disclosure practices aim to address this issue by minimizing the number of attributes that are particularly vulnerable to re-identification. Also, the level of detail is reduced using age categories or only the first digits of the ZIP code, as shown above. A typical goal is to achieve a standard known as k-anonymity. For example, a released dataset satisfies 5-anonymity if at least five records exist for each combination of gender, age, and ZIP code. While this approach likely reduces the hit rate that an attacker can achieve, it is far from solving the problem and fails to provide any reliable privacy guarantee to individuals.

Of course, one can continue by reducing or coarsening the data even further before release. However, this can eventually erode the utility of the released data also for legitimate applications.

> *"Anonymized data isn't."*
> *(Cynthia Dwork)*

Microsoft researcher Cynthia Dwork's quote brings it to the point: "*Anonymized data isn't.*" As long as useful information about individuals is included in the data, it is vulnerable to re-identification attacks (and therefore not anonymous). If we want to provide data that cannot be re-identified, we have to remove so much information that, in the end, the data is no longer useful.

In fact, the risk of re-identification of anonymized data is more significant than the above example suggests:

1. Besides the obvious demographic attributes (e.g., age, gender, ZIP code), other information can be used for linkage attacks (for instance, diagnostic codes in statistics from other hospitals referring to the same individuals).
2. At the time of publication, it is unknown which information about an individual becomes available in the future and might then be exploited for linkage attacks. That said, once released, it is de facto impossible to revoke the data from the internet.
3. Attackers can rely on growing databases about individuals and sophisticated re-identification techniques, which further erodes the effectiveness of current anonymization practices. The next chapter shows how attackers might exploit statistical summary data to reproduce parts of an original dataset.

A spectacular example of a sophisticated re-identification attack took place in the context of the "Netflix Prize" competition. In 2006, Netflix launched a public competition to improve its movie rating algorithm and therefore published an anonymized dataset of their subscribers' movie ratings. As shown in the following figure, the dataset did not include any demographic data or user identifiers (except for a user ID for being able to combine the ratings within the dataset).

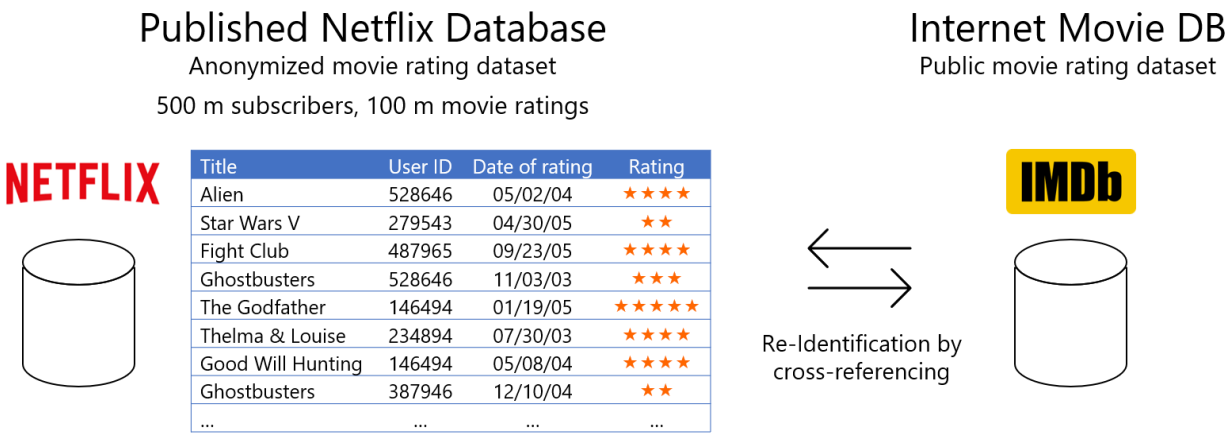


Figure 3: Re-Identification Attack in the Context of the "Netflix Prize" Competition

A few weeks after release, researchers from the University of Texas at Austin have shown that they could uncover many customers' real names from the anonymized dataset.[2]

The attack was performed by linking the entries to corresponding ratings of the public IMDb database, where some of the Netflix subscribers also provided ratings using their real identity. The key for

combining the records was the ratings themselves, although there were many gaps and even inconsistencies between the entries the same person made in both databases.

The researchers concluded that if an attacker knows the approximate date when a subscriber rated six movies, they would be able to identify him or her in 99% of the attacks. Even when the attacker's knowledge about the time of rating and the score is imprecise and totally wrong in some cases, a significant hit rate can still be achieved.

Another disturbing data linkage attack was performed using the public New York City taxi dataset, which contains data about yellow cab rides, including date and time of pick-ups and drop-offs, number of passengers, destination, and fare amount. It does not contain any personally identifiable information about the passengers. However, it is relatively simple to link this data to information made available with paparazzi photos published on gossip websites. By then linking GPS information, it was easy to find out the celebrities' destination addresses, the amount of fare, and the tips that the VIPs had given.

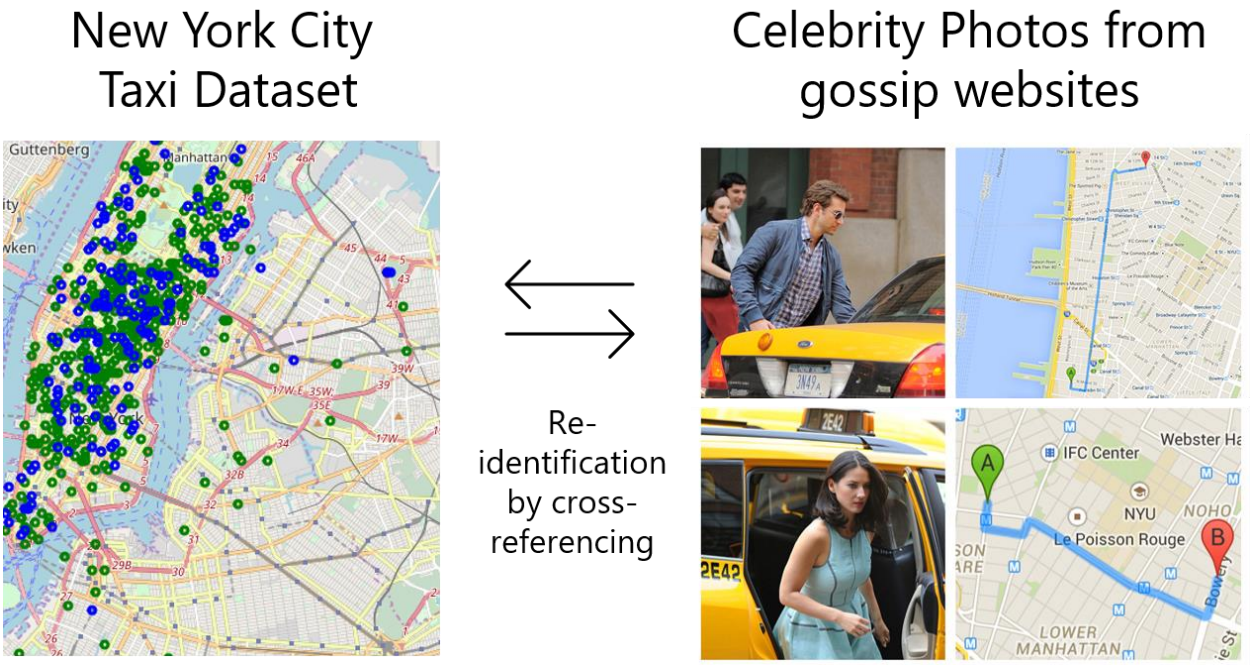


Figure 4: Re-Identifying Celebrity Taxi Ride Information Image sources: medium.com, gawker.com

The Differential Privacy Concept

The critical ambition behind Differential Privacy is that no harm should be done to any person because their record is part of statistical analysis. An example of economic damage caused by private data leakage might occur if an individual (let's call him Benjamin) agrees to participate in a medical study. For instance, the findings of the study could include the detection of pre-existing health conditions. If a potential employer becomes aware of this medical information, it could cause them to favor other candidates' applications over Benjamin's.

> *"Computer science got us into this mess. Can computer science get us out of it?"*
> *(Latanya Sweeney)*

Differential Privacy aims to enable deriving general insights from the statistical analyses (e.g., the general correlation between smoking and the risk of lung cancer) while at the same time reliably protecting the personal data of Benjamin (and the other participants) on an individual level.

So what would be a perfect way to prevent potential leakage of Benjamin's record?

Not participating at all! The "opt-out-scenario" is an essential point of reference in the Differential Privacy concept. Let us consider two datasets by way of example:

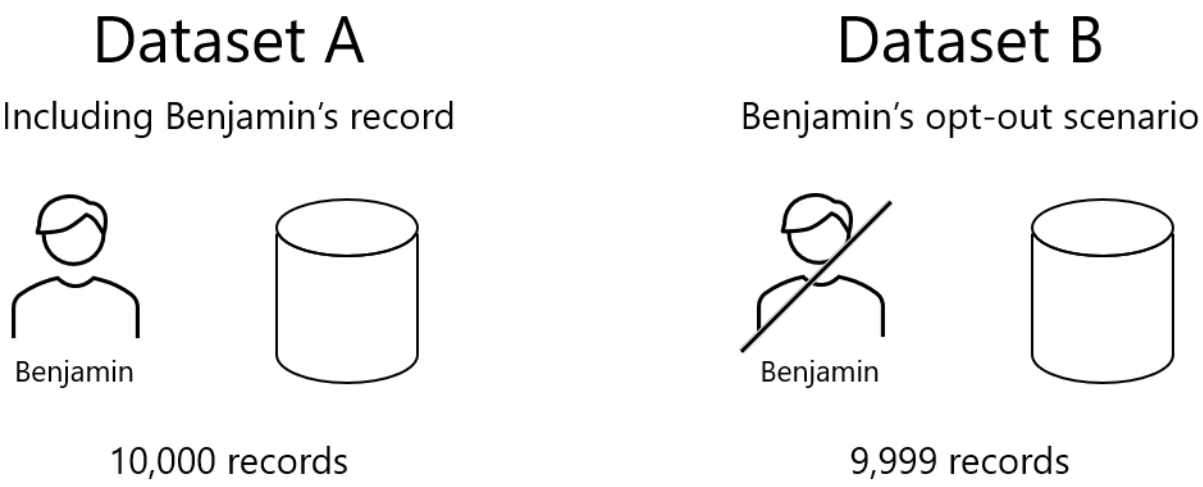


Figure 5: Original Dataset and Opt-Out Scenario

The only difference between both datasets is that dataset B does not contain Benjamin's record. Therefore, it is impossible that analyzing this data results in specific insights about Benjamin.[1]

Differential Privacy requires that any analytical results on dataset A (including Benjamin's record) are identical (or at least very close) to the opt-out scenario results. Indeed, the distinction between both cases is the reason for using the term *Differential* Privacy.

Differential Privacy aims to mask the contribution of Benjamin's record by adding a precisely tuned amount of random noise to the data. Let us take a simple example known as the randomized response

[1] We might derive general insights about the world that can also be attributed to Benjamin without him participating: If the study reveals that smoking causes lung cancer and if we find out that Benjamin is a smoker, we can conclude that he has an increased risk of getting lung cancer. That said, this risk exists regardless of Benjamin's decision whether or not to participate in the study.

technique in market and social research. This time, we want to learn about taxpayers' compliance by asking the participants if they have ever evaded taxes. There is a strong motivation for tax cheaters to hide their real behavior, so they appear to meet social norms. Also, leakage of this information could have severe legal consequences.

What can we do to guarantee the privacy of the participants' sensitive data? And specifically, how can we ensure that no harm might occur to participants of the survey (not even to the tax cheaters)?

We can add random noise at the data acquisition stage of our study by asking the participants to make their answers dependent on coin tosses. To guarantee privacy in this setting, no one but the participant may be aware of the result of the coin tosses.

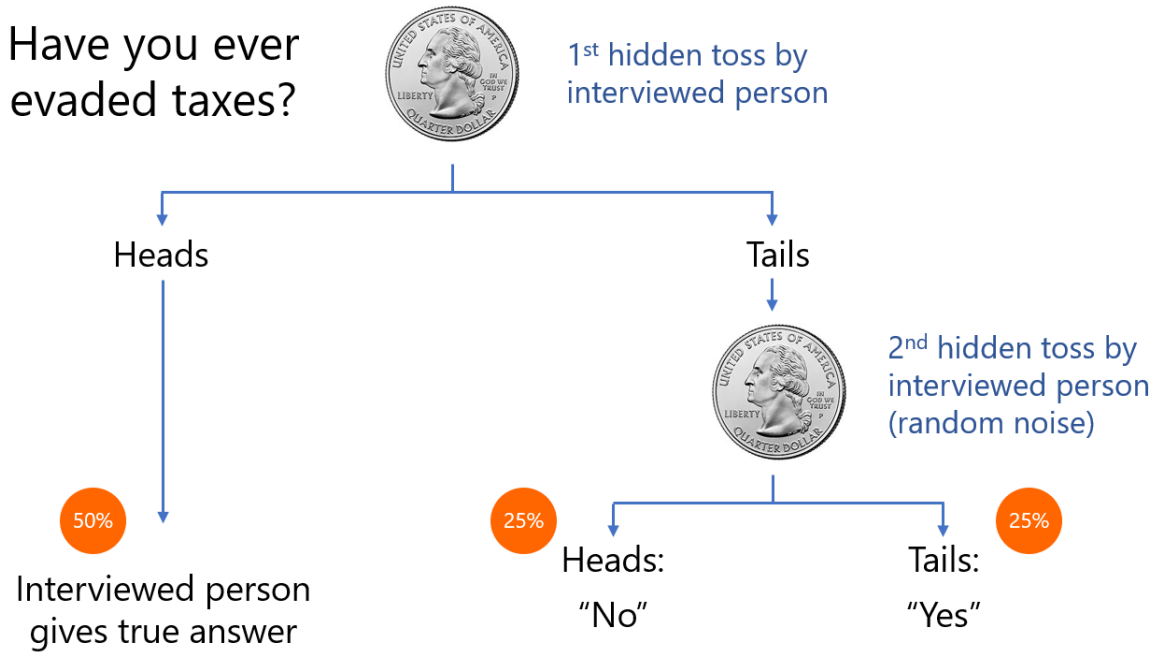


Figure 6: Randomized Response Technique

This method lets the participant toss the coin and answer truthfully if the result is "heads." In case the result shows "tails", then the participant flips the coin for a second time and provides a random answer based on the outcome. As mentioned, no one but the participant is aware of the result of the tosses. No other stakeholder knows if the provided answer corresponds to the truth or not.

Assuming the design is convincing enough for the participants to comply with these rules, we can calculate the relationship between reported and actual tax evaders as follows:

$$Reported_{yes} = 75\% * True_{yes} + 25\% * True_{no}$$

We expect that 75% of the answers from actual tax evaders to be "Yes" (50% truthfully answered plus 25% by chance). Additionally, 25% of tax-compliant respondents give a wrong answer of "Yes" because of their second coin flip (false positives). We can rewrite the formula to estimate the actual share of tax evaders depending on our survey results:

$$True_{yes} = 2 * Reported_{yes} - 0.5$$

For instance, if 45% of the participants responded, "Yes," we expect that 2 * 0.45 – 0.5 = 40% of the participants have indeed evaded taxes.

Due to the introduction of random noise, we face statistical imprecision of the result. It turns out that this error can be reduced by increasing the number of participants in the study.

In practice, noise is introduced computationally using more sophisticated algorithms than our simple example. Noise can be added at various process steps: during data acquisition, data aggregation, analysis, and release of the results. We will show various examples in the following chapters.

The amount of noise that is introduced to the computation must be chosen carefully. On the one hand, higher quantities of noise increase the level of privacy. On the other side, it is more difficult to derive reliable statistical results when the noise level is too high. There is a tunable knob available to adjust the amount of noise in the trade-off between privacy and utility. This knob is known as the privacy parameter epsilon (ε). It is also called the privacy budget.

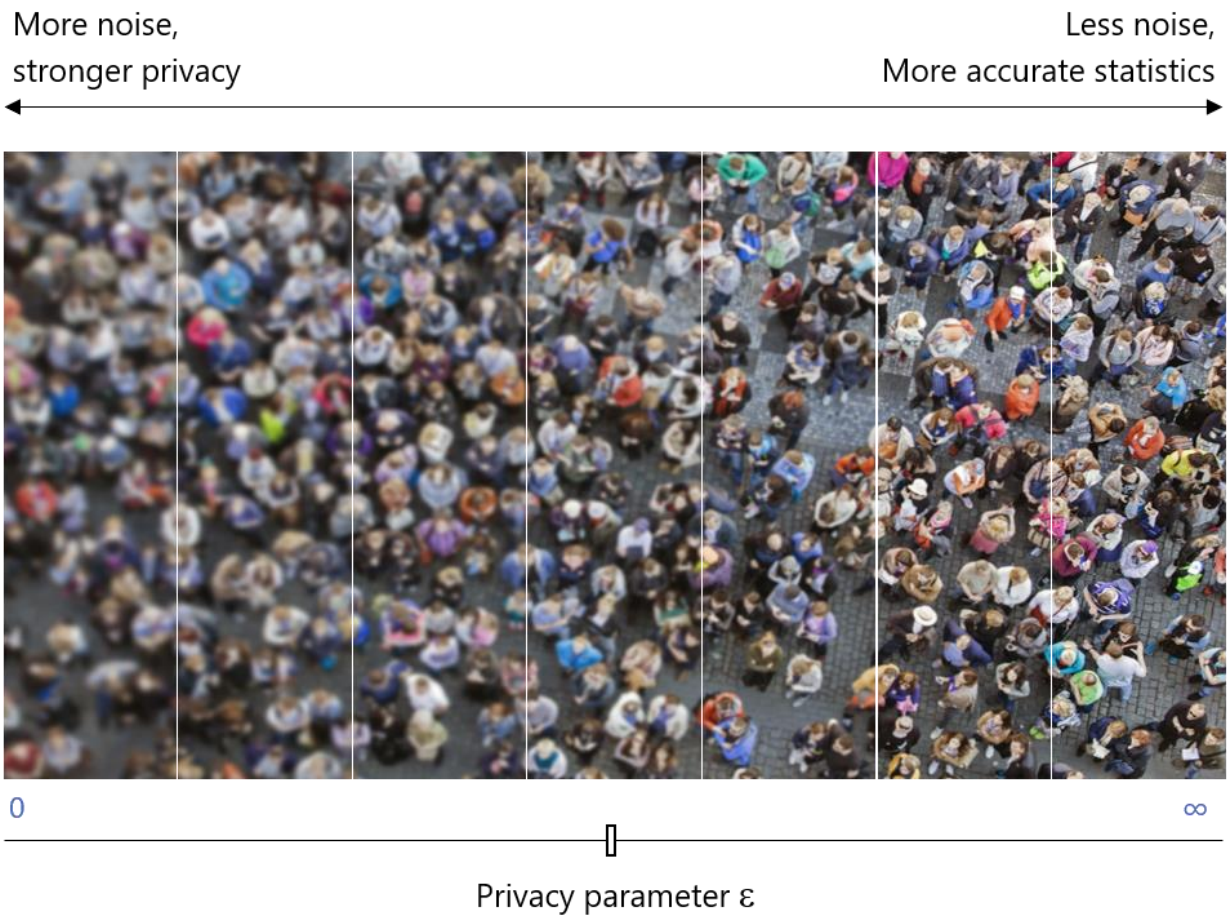


Figure 7: Tuning the Amount of Noise Using the Privacy Parameter ε

Note the inverse relationship between epsilon and privacy: *small* epsilon values lead to *higher* levels of protection and vice versa.

Differential Privacy Applications

Differential Privacy supports a broad spectrum of analytical use cases ranging from generating basic statistics (as seen above) to complex machine learning and deep learning applications, as we demonstrate in these chapters. Here is an overview of typical applications:

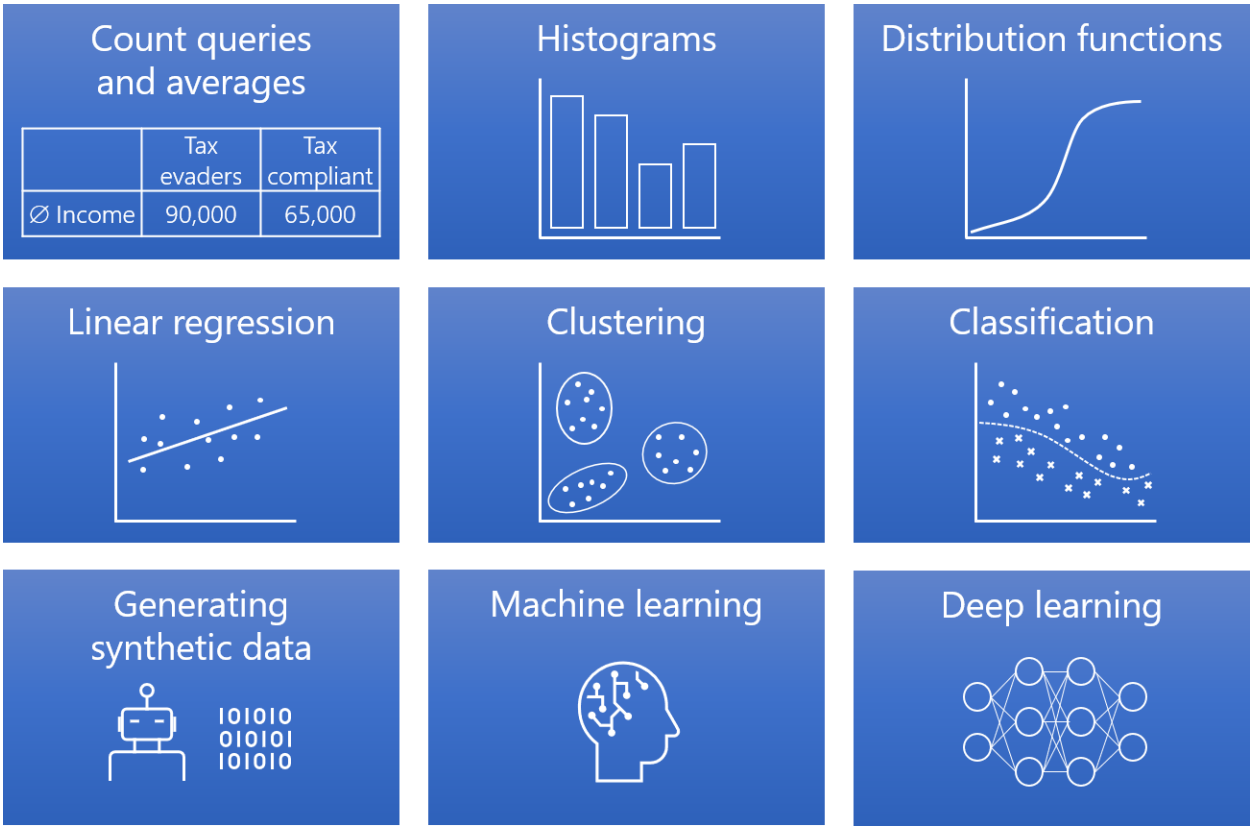


Figure 8: Examples of Differential Privacy Applications

With roots going back to the 1960s and its formal introduction by Cynthia Dwork et al. in 2006, Differential Privacy benefits from a mature scientific foundation today.

It is also gaining significant traction in practical implementations beyond academia. It seems natural that the big tech companies belong to the pioneers, who implement Differential Privacy in large-scale production use cases.

To keep customers' devices secure and up to date, tech companies use diagnostic data to gain insights about the quality and user experience. As an example, Microsoft uses Differential Privacy to protect diagnostic data on Windows 10 devices based on what is known as the "local model". In this implementation, statistical noise is added to the local machine's diagnostic information before transmission to Microsoft.

Differential Privacy is only one part of a comprehensive and layered security and privacy protection strategy that also includes security controls, auditing, policies, user consent, data minimization, encryption, and other measures.

As one of the early adopters, the Windows team benefited from working with Microsoft Research, who invented the Differential Privacy concept with their associates. The current implementation that

protects diagnostic data uses elements of SmartNoise and contributes enhancements back to the open source platform so that these innovations become available to the public.

Other tech companies like Apple and Google use Differential Privacy for similar use cases. These examples benefit from high amounts of data thanks to their extensive device bases. For example, Windows 10 is used on more than one billion devices today, and therefore, the accuracy loss caused by Differential Privacy is negligible.

The US Census Bureau introduced a high-profile milestone for Differential Privacy and decided to use the concept to improve the protection of personal and household data for the 2020 census.

The bureau's track record of implementing organizational and technical measures to better protect personal data goes back to an early census in 1840. Since 2000, the primary technique used to prevent disclosure of individual data is "household swapping".

However, given the availability of increased computational resources and large datasets that might be exploited for reconstruction attacks, the bureau decided to look for a better alternative. It announced to use Differential Privacy for the 2020 census.

Differential Privacy Considerations

Differential Privacy is superior to traditional disclosure limitation techniques in many aspects. It is currently the only technique with a formal privacy guarantee that scales even to subsequent analyses. Researchers argue that Differential Privacy can influence data protection practices to a similar extend as public key infrastructure (PKI) influenced cryptography.

Individuals will likely be more willing to provide personal data (e.g., participating in surveys or medical studies) if a comprehensible privacy guarantee is given.

The same applies to organizations that own personal data. Due to the risk of privacy breaches that might affect an organization's reputation or lead to severe legal consequences, organizations are very cautious about sharing or collaborating on personal data. However, there are societal opportunity costs caused by this reluctance. Many organizations do not own enough data to derive relevant insights (e.g., about rare diseases). Being able to collaborate on such data with other entities would be highly beneficial for the society.

One problem of Differential Privacy is that the concept is less intuitive and more difficult to explain. Convincing analytics and machine learning practitioners can also be challenging since their work results will be less accurate in many cases. Other stakeholders might find it disturbing to accept the distortion of statistical results – especially if the data for producing more accurate results is available.

However, the alternative to Differential Privacy will not be an unprotected release of data in most cases. The data owner or privacy laws might require extensive use of traditional disclosure limitation techniques if no other protection is available. Consider how much information might have to be removed to satisfy 5-anonymity for all records in a database. The resulting loss of the utility of statistical analyses will be substantial in many cases.

Furthermore, the accuracy impact due to the noise can be compensated by adding more data, as we will investigate in the next chapters. In fact, the regularization techniques used in Differential Privacy can also benefit machine learning scenarios since the models tend to generalize better to unseen data.

The experience with Differential Privacy usage in the 2020 census in the United States with regard to effectiveness and stakeholder acceptance will probably be crucial for its adoption in the upcoming years.

We see that stakes for data privacy become higher due to severe, large-scale breaches, growing public awareness, and increased regulations like the GDPR. This might be a strong driver for a rigorous and formal privacy protection framework like Differential Privacy.

Differential Privacy might help implement the GDPR right of individuals to request that their records are deleted. Applying this requirement to an existing machine learning model is challenging. In practice, it is not known which traces of an individual's record are contained in the trained model. The model would effectively have to be retrained, just without the data of the individual, who exercised his or her GDPR right. As we have seen, Differential Privacy is based on this opt-out scenario by masking the contribution of individual data points. We shall see if jurisdiction accepts the Differential Privacy concept to satisfy the “right to be forgotten”.

The SmartNoise System

SmartNoise is jointly developed by Microsoft and Harvard's Institute for Quantitative Social Science (IQSS) and the School of Engineering and Applied Sciences (SEAS) as part of the Open Differential Privacy (OpenDP) initiative.

The project aims to connect solutions from the research community with the lessons learned from real-world deployments to make Differential Privacy broadly accessible.

SmartNoise is designed as a collection of components that can be flexibly configured to enable developers to use the right combination for their environments.

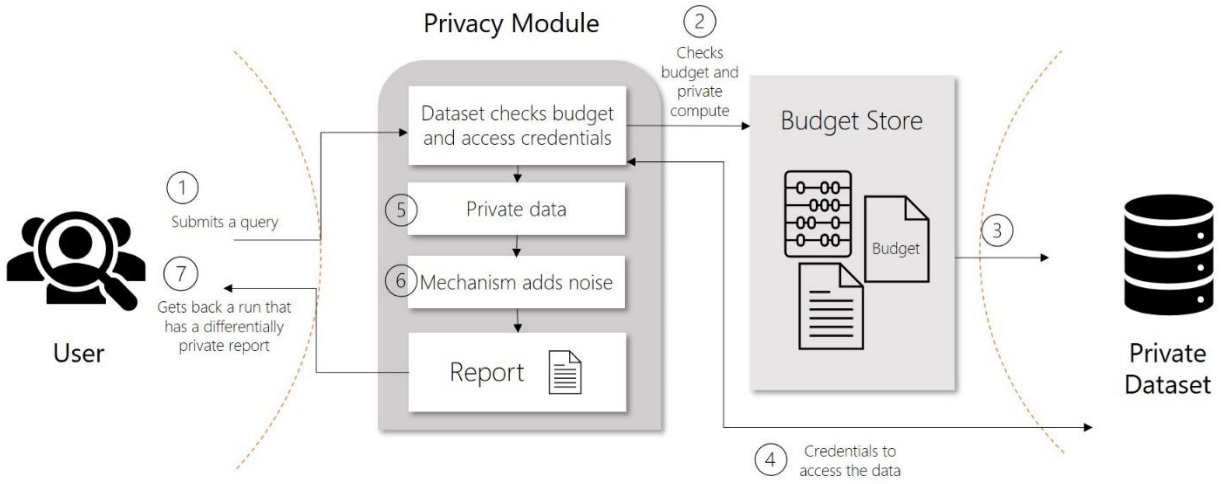


Figure 9: The SmartNoise System

The SmartNoise tools primarily focus on the "global model" of Differential Privacy, as opposed to the "local model." In the global Differential Privacy model, a trusted data collector is presumed to have access to unprotected data and wishes to protect public releases of aggregate information.

The core library includes implementations of the most common mechanisms and statistics and some utility functions like filtering, imputation, and others. It has an interface for composing operations in an analysis graph that can be validated for privacy properties. This validator is the critical functionality, as it ensures users are providing formal privacy assurances.

The core library is designed to be pluggable, allowing the inclusion of new algorithms, including both the "aggregate and add noise" and "sample and aggregate" approaches. The primary releases available in the library and the mechanisms for generating these releases are shown below.

Statistics	Mechanisms	Utilities
• Count • Histogram • Mean • Quantiles • Sum • Variance / Covariance	• Gaussian • Analytic Gaussian • Geometric • Exponential • Laplace • Clamping	• Cast • Clamping • Digitize • Filter • Imputation • Transform • Synthesizers to generate differentially private versions of datasets

On top of the core library, the platform has an SQL data access layer that allows users to compose analysis graphs using the SQL language. This makes it easy to port over existing dashboards and analytics workflows to be differentially private. The SQL language is a limited subset of the operations that can be specified in the full analysis graph.

The SQL data access layer transparently intercepts calls to backend databases that support SQL-92, applying Differential Privacy before returning results. The library includes support for SQL Server, PostgreSQL, Spark, SQLite, Pandas, Dataverse, and Presto.

Because this system's SQL support functions as a data access layer, it is straightforward to add support for other backends, such as Oracle or DB2. The platform provides interfaces that others can extend, and more databases will be added in the future.

To ease deployment, the platform includes a sample hosted service that shows users how to compose heterogeneous queries over the same dataset, fronted by a REST-based endpoint. Users learn how this can be hosted in a cloud environment backed by the open-source MLflow execution environment, and how to track combined epsilon across queries that use IBM's Differential Privacy libraries and a differentially private SGD using PyTorch. This sample service keeps track of cumulative privacy cost only but could serve as a starting point for something that prevents query after the budget has been spent.

Finally, the platform provides a stochastic evaluator tool that can drive any black box privacy algorithm to test for adherence to privacy promises and checks the accuracy and bias.

Protecting against Privacy Attacks

Reconstructing Original Data from Published Statistics

Let us play an attacker's role trying to reconstruct sensitive information of a demographic dataset based on published summary statistics. In a second step, we use SmartNoise to create a differentially private version of the summary statistics and attempt to perform the attack again.

The dataset is a simplified sample of US residents' demographic records from the Public Use Microdata Sample (PUMS) statistics. It contains 1,000 records with the following information: Education, ethnicity, age, gender, marital status, and income. Our attacker has only access to the published summary statistics of the income, which is considered sensitive information that he aims to recover.

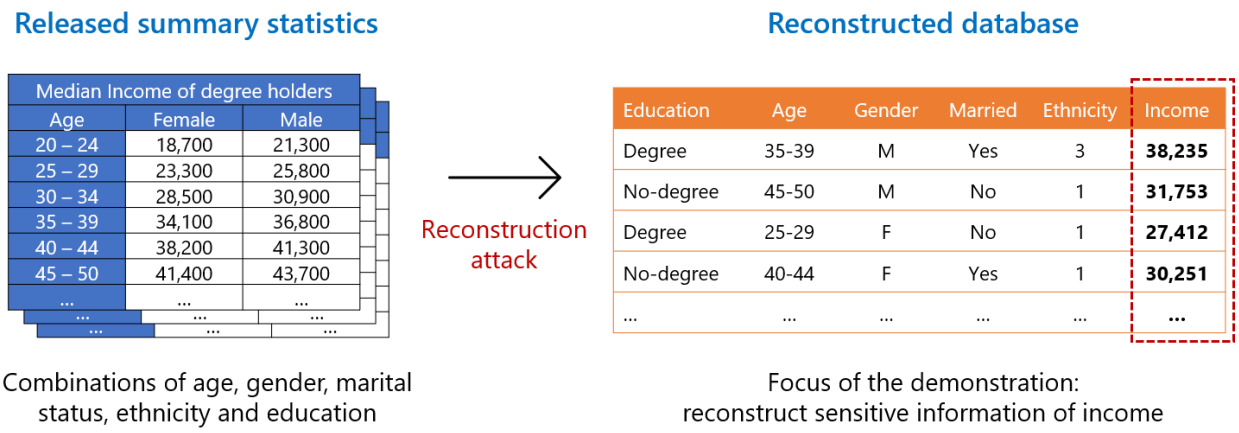


Figure 10: Anatomy of a Database Reconstruction Attack

Protection is provided to the released information by statistical disclosure limitation similar to the k-anonymity principle explained above: No information (means or medians) is released for the subgroups (combinations of education, ethnicity, age, gender, marital status) when the size is smaller than five records (similar to 5-anonymity).

The reconstruction attack approach is relatively straightforward: We aim to generate a dataset containing 1,000 records that is consistent with the published summary statistics. This is basically a combinatoric task with a significant complexity level due to the many possible variable interactions. However, in principle, it is nothing more than solving a giant puzzle.

We are using the Z3 Theorem Prover from Microsoft Research to accomplish this task. Z3 is used to solve mathematical equations, puzzles, games like Sudoku, and other combinatoric problems.

For the reconstruction attack, we must consider a couple of issues. Firstly, Z3 is probably able to reconstruct a part of the database, but not all records. The amount of correctly reconstructable data depends on the quantity of the released information and its detail level. Secondly, there is typically more than a single way to reconstruct the data that satisfies the summary statistics' constraints. Thirdly, an attacker might leverage additional information from outside the statistical release about individuals in the database to improve the attack results.

The complementary Jupyter notebook 1-reconstruction-attack includes several attack scenarios that can be tuned to experience the effect of additional data, changing the level of disclosure limitation and further adjustments.

In our case (minimum of 5 records per subgroup, no additional information leveraged, the statistical release includes average and median), we get the following results:

- 120 (12.0 %) incomes reconstructed exactly
- 257 (25.7 %) incomes reconstructed within $2,000
- 380 (38.0 %) incomes reconstructed within $5,000

Note that we can assess the attack's quality since we have access to the original dataset for comparison. A real attacker would not have this luxury.

So, how can we use SmartNoise to successfully protect the original data against such re-identification attacks? We generate differentially private versions of the summary statistics with a privacy parameter of 1.5 to find out. The results show that this privacy level is sufficient to prevent the reconstruction attack. Z3 cannot generate a dataset that is consistent with the differentially private releases of the summary statistics. The random noise caused too many contradictions to generate a matching dataset.

We have discussed that the price of achieving this level of privacy protection is some loss in the accuracy of the released data. The following diagram provides an impression of this effect by comparing the income distribution of the original and differentially private release:

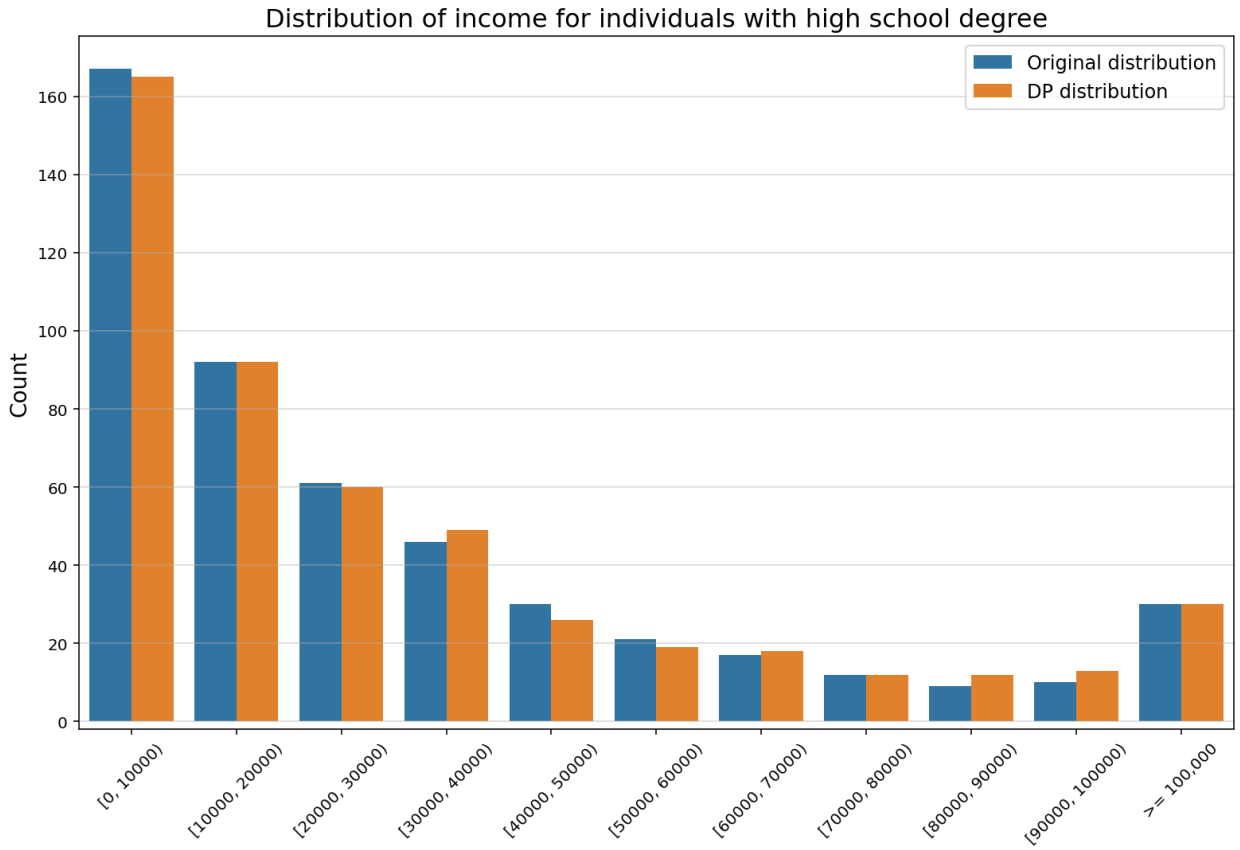


Figure 11: Income Distribution of Original Data Compared to Differentially Private Version (ε = 1.5)

Even though the database contains only 1,000 records, both histograms look relatively similar when a privacy budget of 1.5 is used. We investigate the effect of dataset size and privacy level on the accuracy in more detail in the next chapters.

Re-Identification Attack

In this section, we go one step further by trying to re-identify individuals from a medical dataset. The approach is similar to the discussed examples of revealed Netflix subscribers and New York taxi passengers.

Our showcase compares the effectiveness of anonymization and Differential Privacy. The complementary Jupyter notebook 2-reidentification-attack can be used to experience these scenarios in action. The flow of the demonstration is as follows:

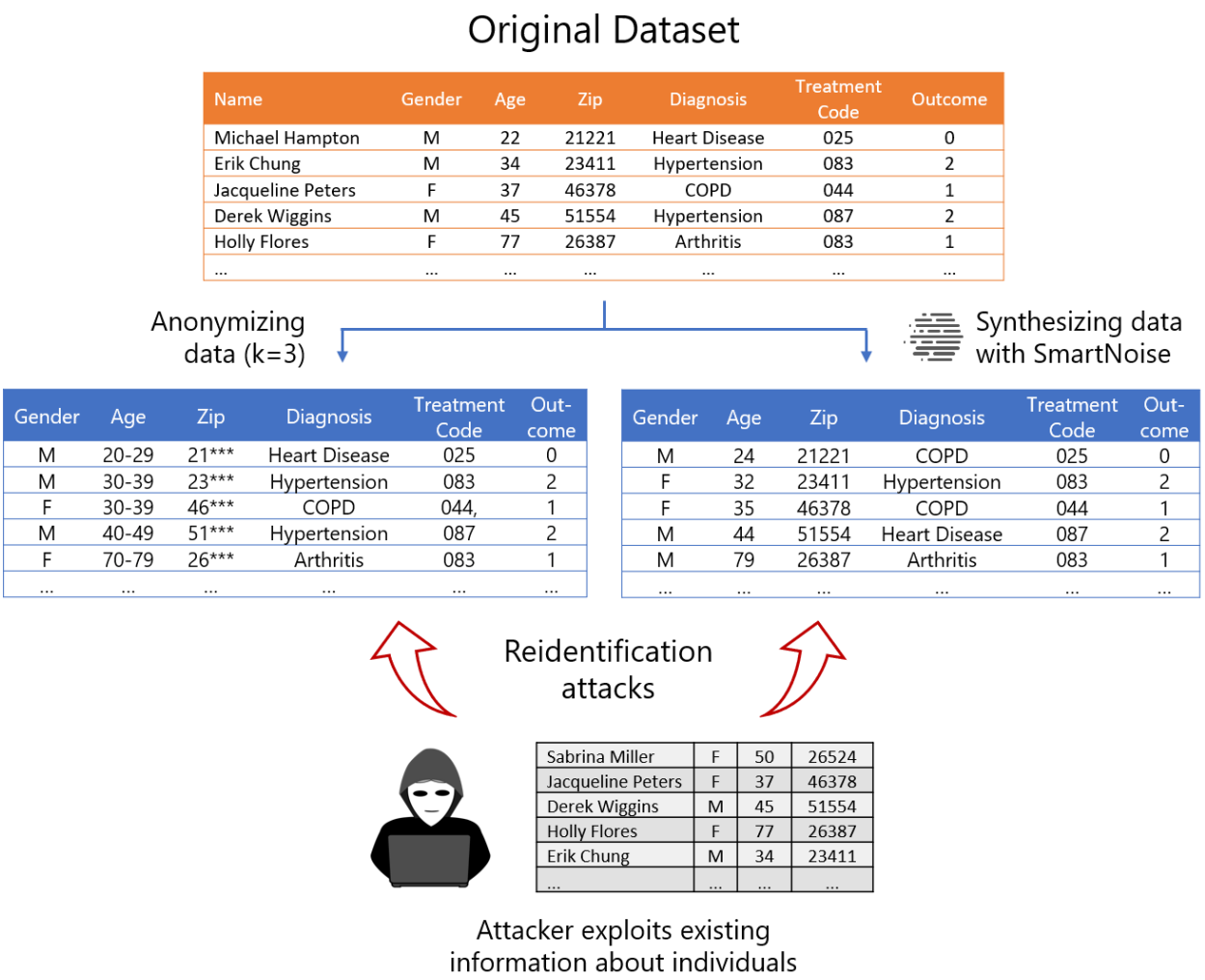


Figure 12: Comparing the Privacy Protection of Anonymization and Differential Privacy

Our starting point is a dataset that we have created using artificial data to simulate medical records for the demonstration. We have generated 30,000 patient records containing names, ages, zip codes, diagnoses, treatment codes, and medical procedures outcomes.

To simulate the attack against an anonymized dataset, we redact the patients' names and coarsen the demographic information: ages are reported as ten-year ranges, and zip codes are shown only to the first two digits. The resulting dataset is 3-anonymous, which means that for each combination of the reported "insensitive" variables, gender, age, and zip code, at least 3 matching records exist.

In our scenario, the attacker has access to a dataset of records with demographic information that he or she uses to identify individuals from the medical dataset. We have derived the attacker's data collection and the medical dataset from the same source for our demo. This approach enables us to validate the hit rate of identified individuals after the attack. In a real-world scenario, the attacker would likely leverage various kinds of available information ranging from public datasets to databases about individuals offered on the black market.

Since we do not use additional information about the patients that potentially exist (e.g., pre-existing conditions), we expect a hit rate of approximately 30%, given that our dataset is 3-anonymous. The results show indeed that we can identify more than 9,000 patients.

The following table shows examples of successfully re-identified patient records:

Patient	Gender	Age	Zip	Diagnosis	Treatment	Outcome	Match
Kimberly Thomas	F	52	78967	Heart Disease	22	unchanged	True
Thomas Foster	M	62	80176	Arthritis	44	unchanged	True
Nathan Huffman	M	59	26129	Osteoporosis	30	intensive care	True
Lisa Bass	F	39	47750	COPD	38	recovered	True
Shawn Bowman	M	80	27472	Alzheimer	35	intensive care	True
Ariel Miller	F	68	92122	Arthritis	36	intensive care	True
Natasha Martinez	F	37	41288	Heart Disease	45	recovered	True
Matthew Wilson	M	43	97748	Heart Disease	48	intensive care	True
Heather Cruz	F	37	55750	Heart Disease	37	unchanged	True
Jeremy Schmitt	M	21	77819	High Blood Pressure	35	intensive care	True
…	…	…	…	…	…	…	…

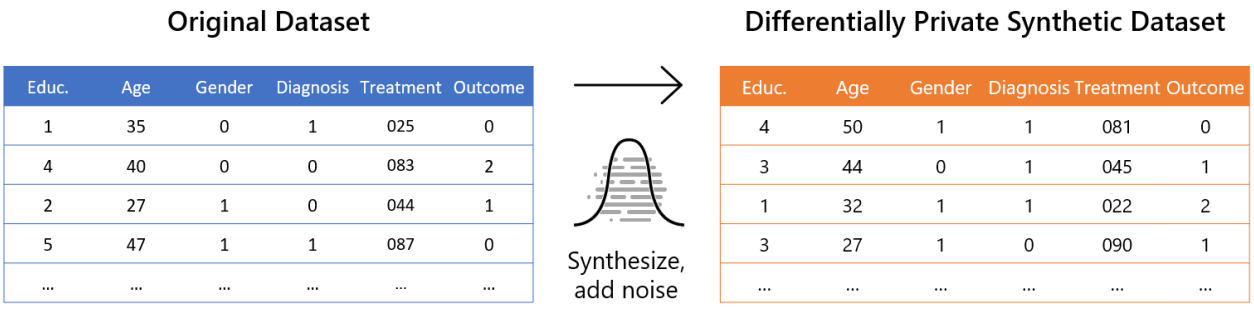
Table 1: 9,016 Medical Records Re-Identified from an Anonymized Dataset

As mentioned, we can count the number of re-identified patients only because we use a unique patient identifier in both datasets. This information is used solely to evaluate the effectiveness of the attack and not for the re-identification itself.

The second re-identification attack goal is to determine if Differential Privacy is a more effective way to protect sensitive medical information in our scenario. Therefore, we use SmartNoise to create a synthesized version of the data based on the original medical data set. Then, we repeat the attack, evaluate its effectiveness, and assess if the synthesized dataset is still useful for statistical analyses.

The generation of synthetic data is a standard statistical disclosure limitation technique aiming to preserve individuals' confidentiality represented in the data. However, its main limitation is the lack of formal privacy guarantees. Let's find out if Differential Privacy can close this gap.

The differentially private synthetic dataset is generated from a statistical model based on the original dataset. While the approach is also available for unstructured data like text or images, the most frequent synthetic data examples are tabular datasets.



Educ.	Age	Gender	Diagnosis	Treatment	Outcome
1	35	0	1	025	0
4	40	0	0	083	2
2	27	1	0	044	1
5	47	1	1	087	0
…	…	…	…	…	…

Educ.	Age	Gender	Diagnosis	Treatment	Outcome
4	50	1	1	081	0
3	44	0	1	045	1
1	32	1	1	022	2
3	27	1	0	090	1
…	…	…	…	…	…

Figure 13: Generating a Differentially Private Dataset Based on Original Data

The synthetic dataset represents a "fake" sample derived from the original data while retaining as many statistical characteristics as possible. The synthetic records shown in the right table are not related to the corresponding rows in the original table.

Various techniques exist to generate differentially private synthetic data, including approaches based on deep neural networks, auto-encoders, and generative adversarial models.

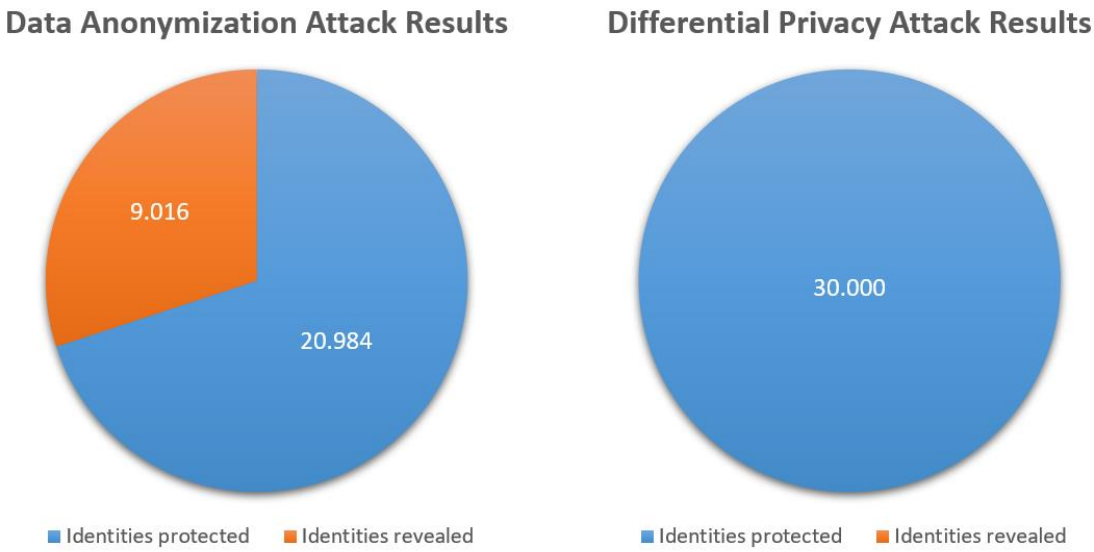
The SmartNoise system includes the following data synthesizers:

Supported Synthesizer	Overview
Multiplicative Weights Exponential Mechanism (MWEM)	• Achieves Differential Privacy by combining Multiplicative Weights and Exponential Mechanism techniques • A relatively simple but effective approach • Requires fewer computational resources, shorter runtime
Differentially Private Generative Adversarial Network (DPGAN)	• Adds noise to the discriminator of the GAN to enforce Differential Privacy • Has been used with image data and electronic health records (HER)
Private Aggregation of Teacher Ensembles Generative Adversarial Network (PATEGAN)	• A modification of the PATE framework that is applied to GANs to preserve Differential Privacy of synthetic data • Improvement of DPGAN, especially for classification tasks
DP-CTGAN	• Takes the state-of-the-art CTGAN for synthesizing tabular data and applies DPSGD (the same method for ensuring Differential Privacy that DPGAN uses) • Specifically suited for tabular data, avoids issues with mode collapse • Extensive training times
PATE-CTGAN	• Takes the state-of-the-art CTGAN for synthesizing tabular data and applies PATE (the same method for ensuring Differential Privacy that PATEGAN uses) • Specifically suited for tabular data, avoids issues with mode collapse
Quail-ified Architecture to Improve Learning (QUAIL)[3]	• Ensemble method to improve the utility of synthetic differentially private datasets for machine learning tasks • Combines a differentially private synthesizer and an embedded differentially private supervised learning model to produce a flexible synthetic data set with high machine learning utility

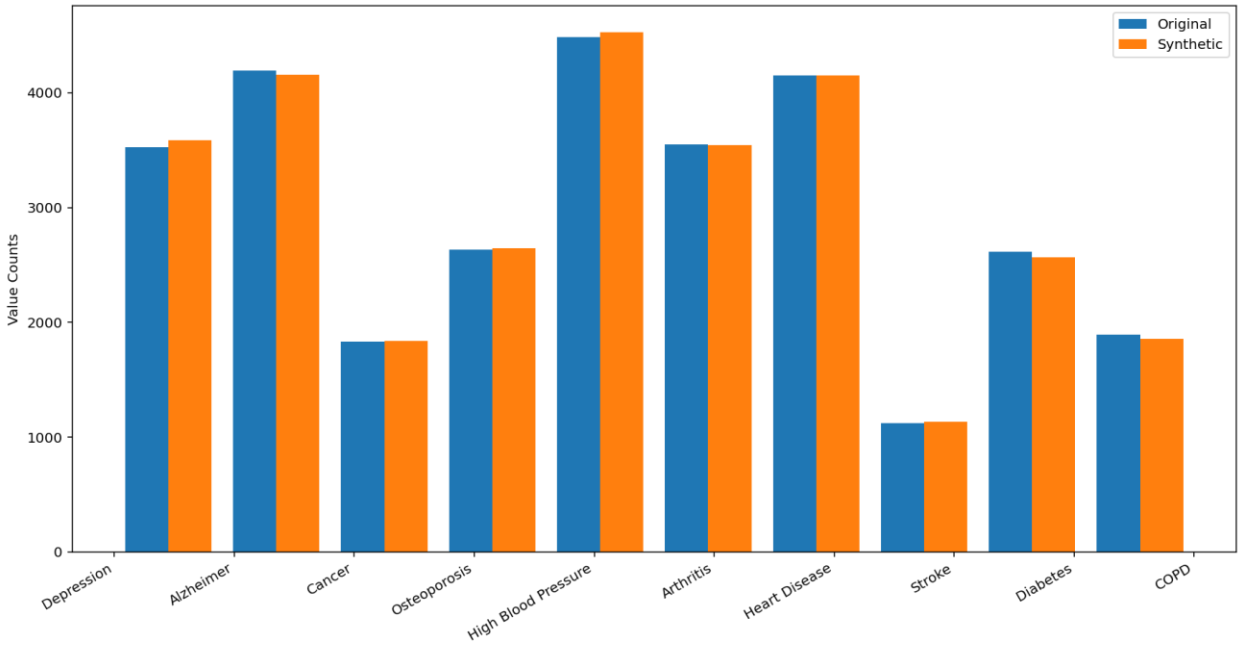
For our scenario, we are using the Multiplicative Weights Exponential Mechanism (MWEM), which is a straightforward yet effective synthesizer. We also need to specify the desired privacy guarantee (epsilon) as an input of the synthesizer process. Here, we chose a level of 3.0, assuming it leads to the right balance between privacy protection and utility for statistical analyses.

The synthesizer does not convert the original dataset. Instead, it generates a new, differentially private sample. Therefore, we lose the ability to validate actual matches based on re-identification attacks on the synthetic dataset. However, we can count the number of *potential* matches the attacker could identify (identical combinations of gender, age, and zip code).

We did not find a single potential match in our tests and can, therefore, safely exclude any actual matches. The following comparison summarizes the results based on data anonymization and Differential Privacy:



The above results confirm the effectiveness of the Differential Privacy concept for protecting confidential data. But we also need to assess the utility for statistical analyses like we did in the previous section. For this purpose, we compare the distribution of diagnoses between the original and the synthesized dataset.



The histogram looks promising: The synthetic dataset apparently represents the distribution of individual variables from the original dataset. We further investigate the relationship between privacy guarantee, dataset size, and statistical accuracy in the following.

The Impact of Privacy and the Amount of Data on Statistical Accuracy

The purpose of this section is to deepen our view of the trade-off between privacy guarantee and statistical accuracy. We have already seen that it is possible to relax the privacy level to improve the

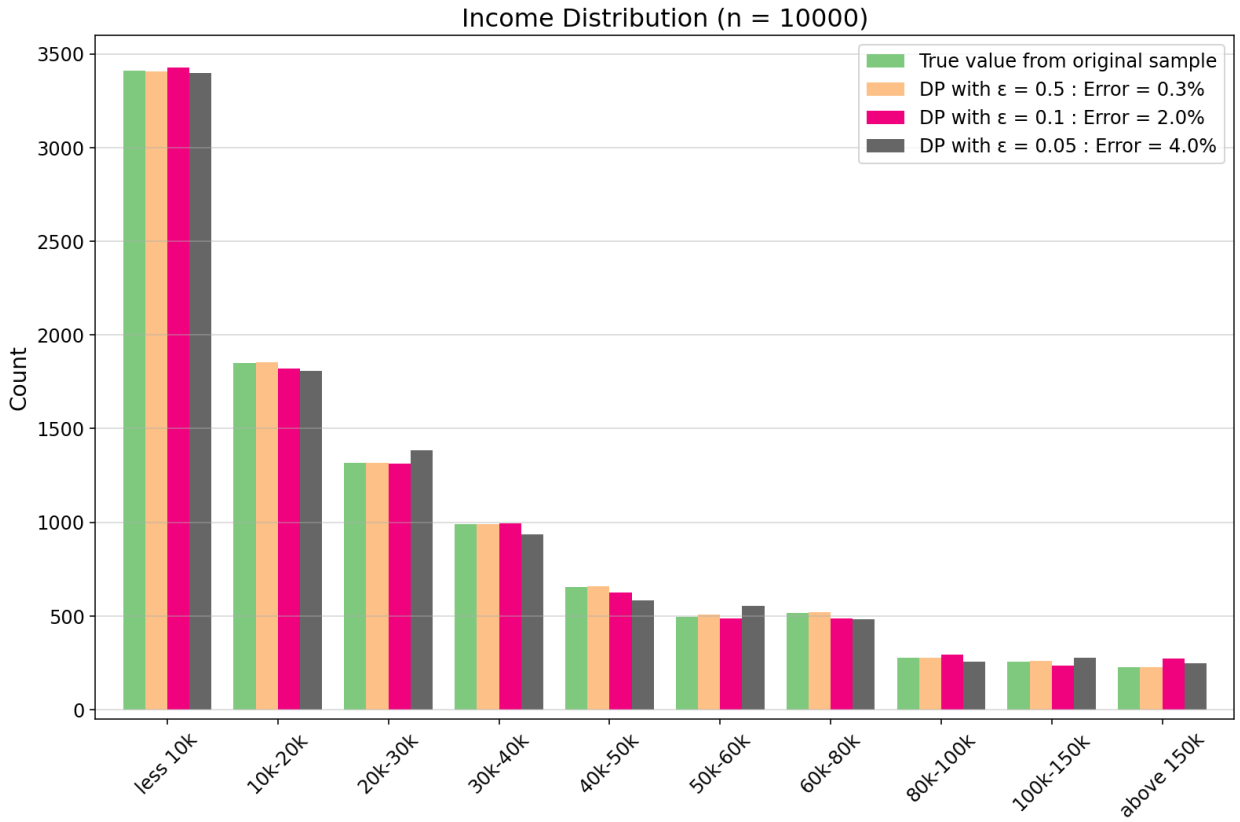
quality of statistical outcomes and vice versa. That said, there is a third factor that we can exploit: increasing the amount of data.

We are using a database containing demographic records about Californian residents from the Public Use Microdata Sample (PUMS) statistics. The dataset includes more than 1.2 million records and is thus suitable to experiment with different sample sizes. The data contains personal information like gender, age, ethnicity, income, and marital status. For this analysis, we focus on creating histograms of the income variable.

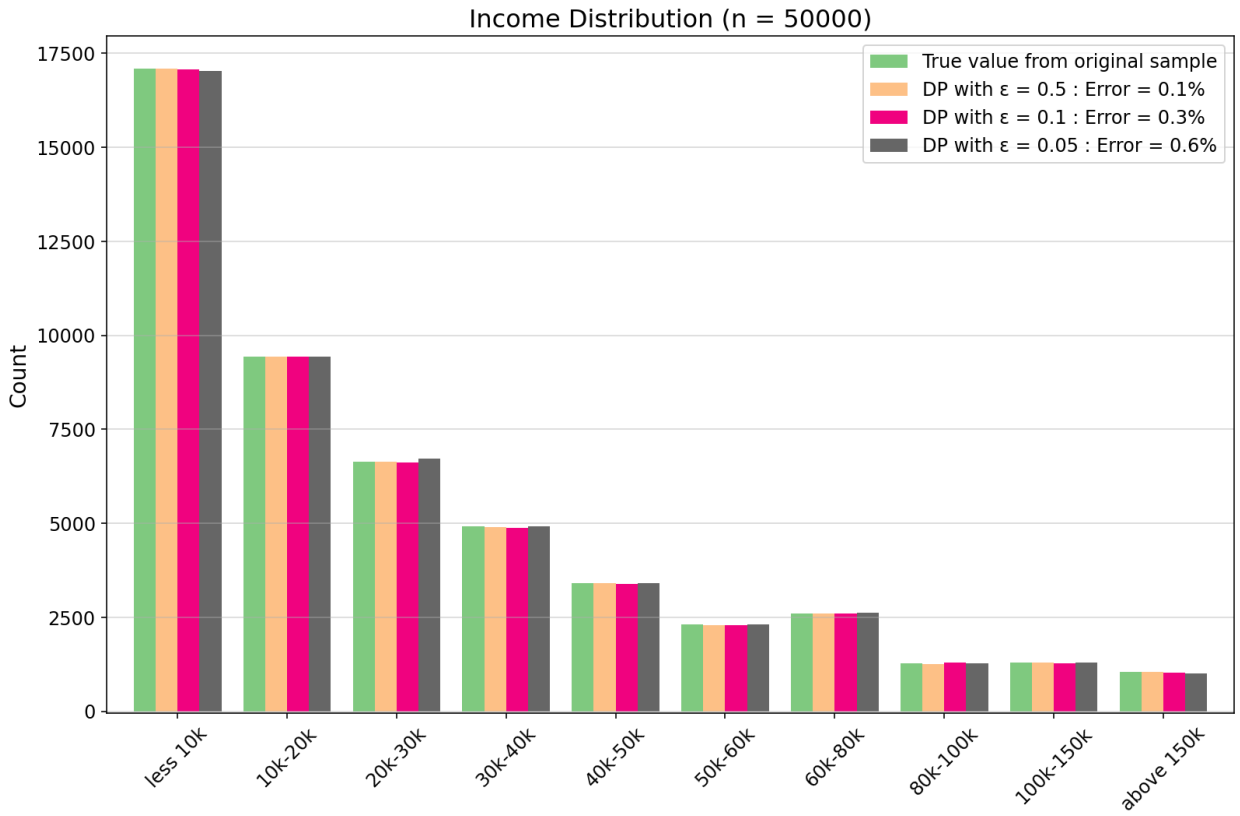
The following illustrations compare histograms of "true" samples from the original dataset to differentially private histograms, each generated at different privacy budgets (controlled by the parameter epsilon). The differentially private histograms were created using the geometric mechanism of the SmartNoise system. The demo scenario is available as a Jupyter notebook 3-histogram-sample developed in Python.

For our smallest sample of 10,000 records, we can see some deviations between the original sample and the differentially private histograms. As expected, the differences increase with more restrictive privacy levels since a higher amount of statistical noise is added (corresponding to lower values of epsilon).

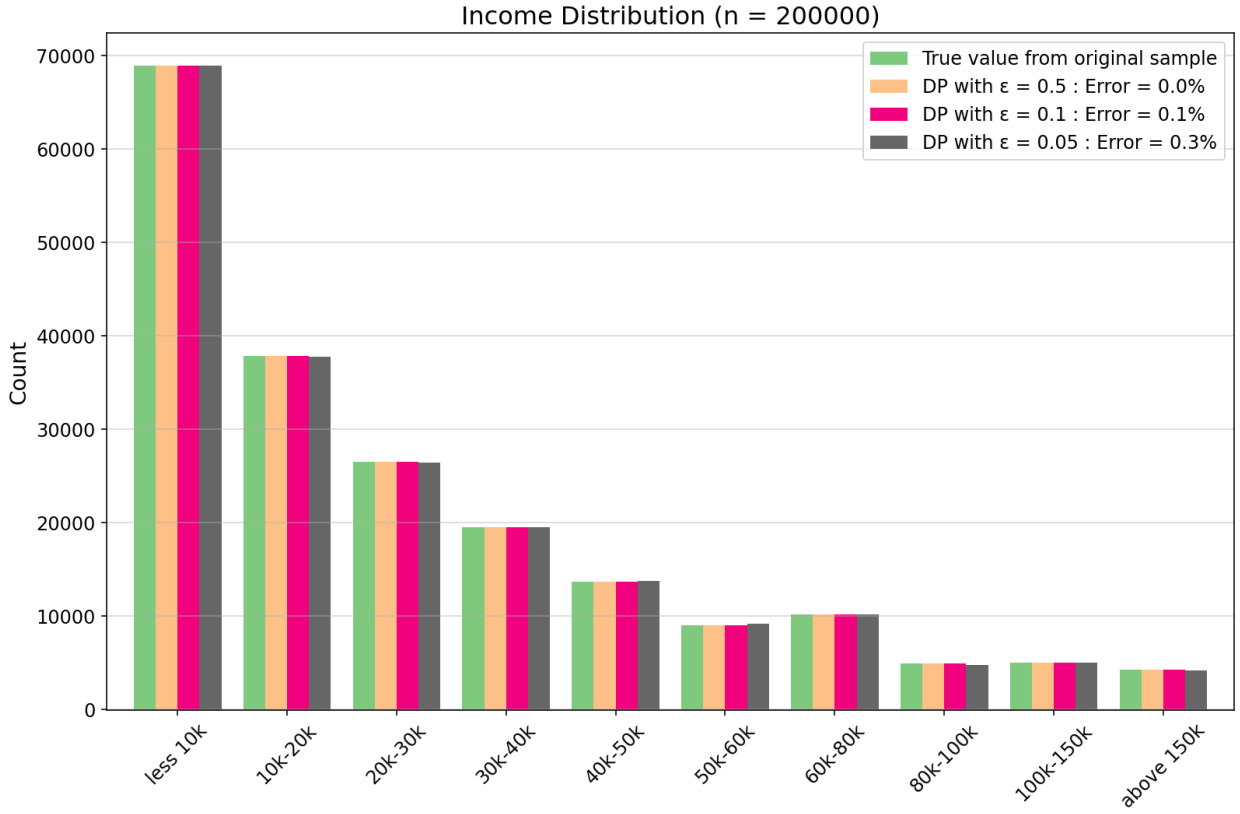
We also have calculated the percentage error to get a measure for the deviation at each privacy level (see legend).



By increasing the sample size to 50,000 records, we can observe that the histograms resemble each other. There are still some differences in our most restrictive privacy setting with an epsilon of 0.05 (e.g., income levels from 20k to 30k). However, the total percentage error is significantly reduced compared to the above example.



The final example is based on a sample of 200,000 records. There are almost no observable deviations between the bars. The overall percentage error shrinks to 0.3% in our most restrictive setting of epsilon equal to 0.05.



The above observations support theoretical findings that the decreasing sampling error tends to offset the statistical noise when more records are added. Big data pays off in differentially private settings.

Differentially Private Machine Learning

In the previous chapter, we have seen that Differential Privacy can generate aggregated statistics like histograms. This chapter discusses more demanding applications of machine learning with structured data. A key question is if the loss of accuracy caused by the added statistical noise can be compensated by increasing the amount of data. Then, in the final chapter, we look at an even more challenging use case of analyzing medical images with deep learning.

In a typical machine learning workflow data is transformed in several steps: First, individual records are collected, then they are aggregated into a dataset, and finally embodied as parameters of the trained machine learning model.

The following illustration shows at which stages Differential Privacy can be applied to the workflow.

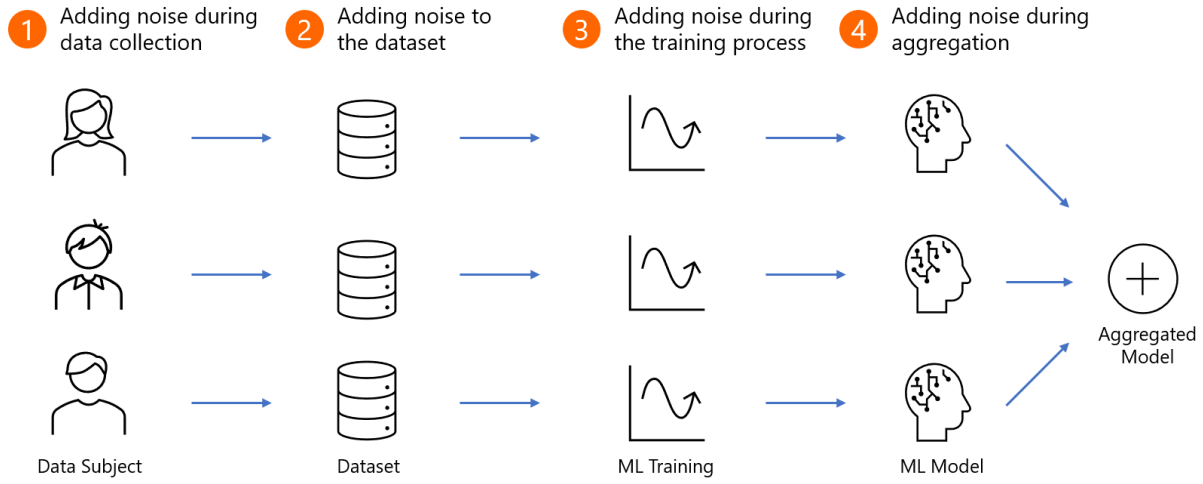


Figure 14: Options to Apply Differential Privacy to a Machine Learning Workflow

A key question in selecting the best approach is which stakeholders should be allowed to access the data in an unprotected state.

We provide examples for adding noise to the dataset before training (option 2) and conducting differentially private training (option 3) in the next sections.

Adding Noise at various Stages of the Machine Learning Process

1. Adding Noise during Data Collection

The most restrictive approach would be to let the data subject add noise during data collection. This corresponds to the discussed method of asking an interviewed person to provide either true or false information based on flipping a coin. This way, nobody else involved in the data collection or downstream machine learning process is aware of the real answer to the question.

Since this approach is impractical in many cases, noise is often added automatically on the device that is used for data collection. We have discussed this approach in the example of protecting diagnostic data of Windows 10 devices.

2. Adding Noise to the Dataset

Noise can also be added to the data at a later stage of the workflow. We have already seen the example of using the SmartNoise MWEM synthesizer to generate differentially private datasets.

One benefit of this approach is that the whole synthesized dataset could be disclosed (even to potentially untrusted parties) without concerns of disclosing private data.

With this method, the span of trust includes the persons who collect the original data and generate the synthetic dataset from it.

One can then perform machine learning and other types of analysis on the synthetic dataset. There are some drawbacks to this approach; e.g., there can be a substantial impact on the model's accuracy depending on the characteristics of the dataset.

Furthermore, the resulting model often needs to be evaluated using test data from the original distribution. Therefore, at least a part of the dataset is kept unprotected.

3. Differentially Private Machine Learning Training

The machine learning process is often based on an algorithm (e.g., Random Forest) or a neural network, which is derived from a training dataset. The result of this process is the trained machine learning model that can make predictions for new data. The related privacy risk is that an attacker might extract information about individuals from the trained model.

Using Differential Privacy with machine learning aims to prevent these kinds of attacks. However, the model is still trained on the original (unprotected) data. Thus, while Differential Privacy protects the resulting model, **the dataset is not protected by this approach.**

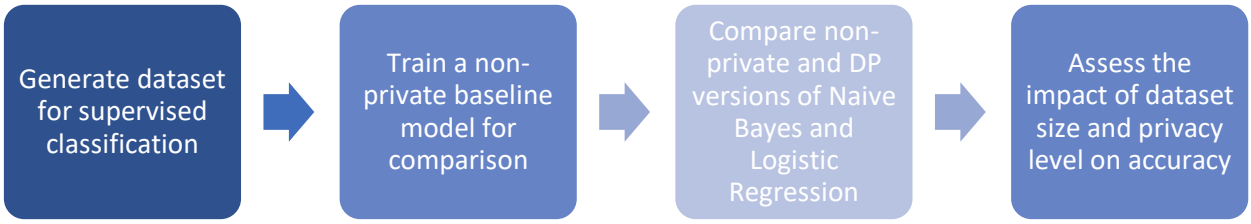
Several machine learning algorithms like Logistic Regression have been adjusted to learn in a differentially private way. The same applies to the way how neural networks can be trained in deep learning.

Differentially Private Machine Learning Algorithms

The following case study applies differentially private machine learning algorithms to a tabular dataset. We are adding noise at the training stage of the machine learning workflow (option 3 as described in the previous section).

The key aspect is to compare the performance of the resulting model with non-private machine learning. Furthermore, we investigate the impact of the dataset size when using Differential Privacy. Can we compensate for the accuracy loss induced by the added noise by acquiring more data?

The example is based on the following workflow:



We have chosen a classification task for our showcase. However, Differential Privacy also enables other supervised tasks (e.g., regression). The demo scenario is available as a Jupyter notebook 4-ml-dp-classifier developed in Python, which you can use to reproduce the results and customize to your use cases.

Generating a Dataset for a Supervised Classification Scenario

We are generating an artificial dataset for this scenario. While it is generally preferred to test new approaches with real-world data, the selected method has some advantages:

- The dataset's complexity can be adjusted to test differentially private machine learning at different levels of difficulty.
- There is no limitation in the dataset size, which is beneficial for our experiment since we want to learn about the effect of various data volumes.

Our experience shows that comparable results can also be achieved with real-world datasets. Furthermore, we discuss a deep learning scenario on a real medical dataset in the next section.

We are using the make_classification function from the popular Scikit-Learn library to generate an artificial multiclass dataset based on the following specification:

Parameter	Value	Comment
Number of Instances	Up to 100,000	Varied between experiments. 80% : 20% Train-test split
Number of classes	10	10 balanced classes
Flipped classes	3%	The fraction of samples which are assigned randomly
Clusters per class	2	2 clusters with a class separator of 5.0
Number of features	20	Varying levels of relevance
Informative features	7	Features that are relevant for prediction
Redundant features	5	Linear combinations of informative features
Repeated features	5	Random copies of informative and redundant features
Useless features	3	Random noise (unrelated to differential privacy)
Feature data type	Float	

Table 2: Specification of the Artificial Dataset for Differentially Private Machine Learning

Approx. 3% of instances are assigned randomly to the classes. Therefore, we expect an optimal classifier to achieve an accuracy score of roughly 97%.

The following illustration of several feature combinations of the dataset provides an overview of the classification task's complexity.

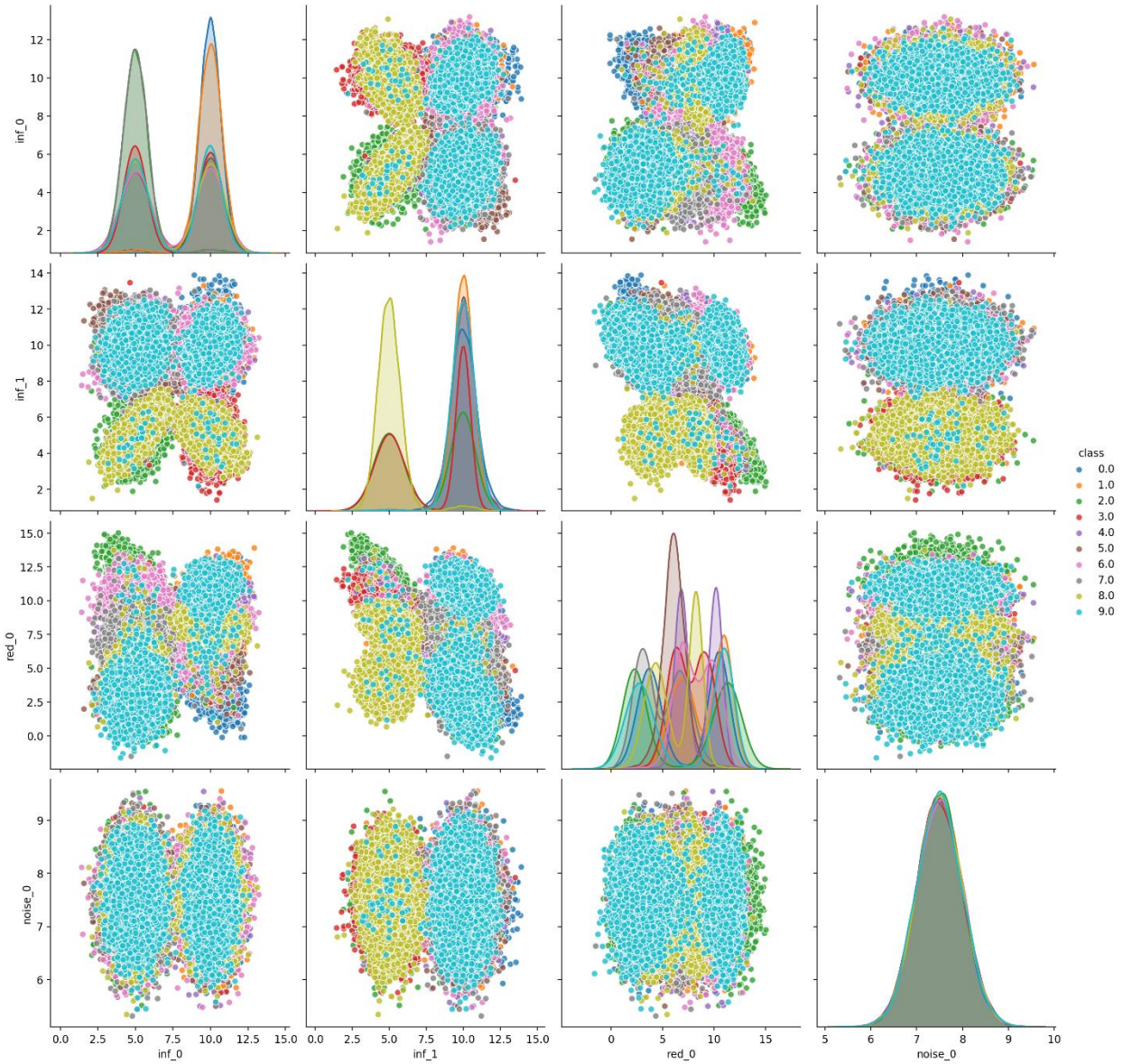


Figure 15: Scatter Plots of Selected Features of the Artificially Generated Dataset

We see pairwise scatter plots of selected feature variables (2 informative, 1 redundant, and 1 feature to introduce noise) and the 10 classes as color-coded points. The figure provides an idea about the complexity of the classification task.

The best constellation for learning to distinguish between the classes is the intersection between both informative features inf_0 and inf_1. We can identify clusters along both axes. On the other side, the intersections between the feature noise_0 and the other variables show that the model will not learn useful patterns from the "noise" features. The structure of the scatter cloud changes along the informative variable axis but is unchanged along the noise dimension.

Training of a Non-Private Baseline Model for Comparison

We use a Random Forest algorithm as a baseline since this classifier typically provides robust performance. Hence, it should indicate the achievable classification metrics for the dataset. Since approximately 3% of the instances are intentionally mislabeled, the maximum accuracy level is around 97%.

Without any further tuning, the trained model achieves the expected classification performance right from the start, as the test set metrics show:

Accuracy score: 0.97 --- ROC AUC score: 0.985				
Classification report	Precision	Recall	F1-Score	Support
Class 0	0.97	0.98	0.98	1999
Class 1	0.97	0.97	0.97	2006
Class 2	0.98	0.97	0.97	2000
Class 3	0.97	0.97	0.97	2006
Class 4	0.97	0.97	0.97	1999
Class 5	0.98	0.98	0.98	1996
Class 6	0.97	0.97	0.97	1993
Class 7	0.97	0.96	0.97	1996
Class 8	0.97	0.97	0.97	2005
Class 9	0.97	0.97	0.97	2000
Macro avg	0.97	0.97	0.97	20000
Weighted avg	0.97	0.97	0.97	20000

Table 3: Non-Private Random Forest Baseline Classifier Test Set Performance

Since we have included a combination of useful, redundant, and useless variables, it is interesting to determine which features were relevant to the Random Forest algorithm to predict the classes.

The feature importance plot shows the contribution of each variable to the classification model. As expected, informative and redundant features are among the top performers (the latter also perform well because they combine information from various informative variables). As expected, the useless features do not provide any insights for the classification model.

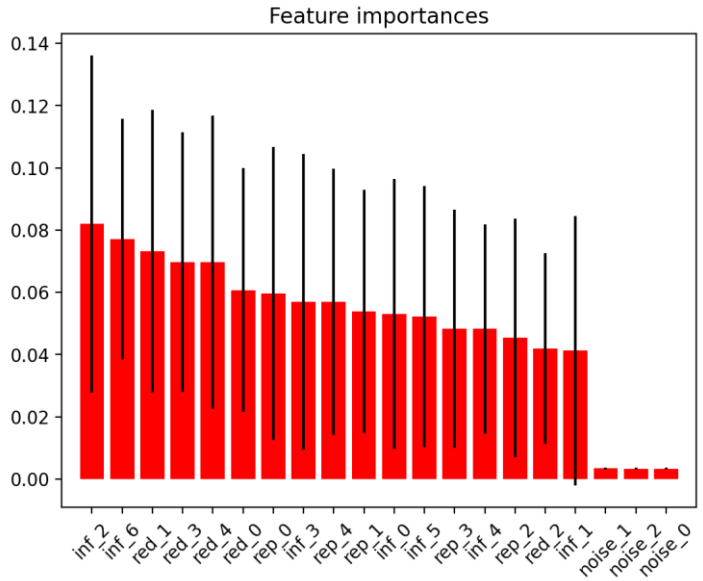


Figure 16: Random Forest Feature Importance of the Artificial Dataset

Compare Non-Private and DP Versions of Naive Bayes and Logistic Regression

In this section, we compare the performance of non-private and differentially private machine learning. Ideally, this would be based on a differentially private version of the Random Forest algorithm. However, the Differential Privacy library used for the comparison does only support Naive Bayes and Logistic Regression algorithms at this time.

For all experiments, the features have been scaled by subtracting the mean and then dividing by the standard deviation using StandardScaler from the Scikit-Learn library.

Naive Bayes

Naive Bayes methods are supervised learning algorithms based on applying Bayes' theorem to assume conditional independence between every pair of features given the class variable's value.

For the non-private experiment, we are using the GaussianNB classifier from Scikit-Learn and perform the training using default values without any hyperparameter tuning. The differentially private version of the Naïve Bayes classifier is taken from the IBM Differential Privacy Library (Diffprivlib). This classifier is based on the Scikit-Learn version and should therefore provide a good reference point for comparison.

We select a privacy parameter epsilon of 1 and specify the minimum and maximum values of the scaled features as required by the classifier (bounds-parameter).

The following table compares the resulting performance of non-private and private training on the test set:

	Non-private training			**DP training** (ϵ = 1.0)			
Classification	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Test n
Class 0	0.70	0.97	0.81	0.74	0.96	0.83	1999
Class 1	0.44	0.20	0.28	0.50	0.44	0.47	2006
Class 2	0.97	0.97	0.97	0.95	0.91	0.93	2000
Class 3	0.91	0.97	0.94	0.92	0.83	0.87	2006
Class 4	0.64	0.76	0.70	0.69	0.63	0.65	1999
Class 5	0.97	0.96	0.96	0.85	0.95	0.90	1996
Class 6	0.70	0.96	0.81	0.70	0.96	0.81	1993
Class 7	0.65	0.95	0.77	0..65	0.93	0.76	1996
Class 8	0.97	0.97	0.97	0.97	0.94	0.95	2005
Class 9	0.72	0.07	0.12	0.61	0.08	0.14	2000
Macro avg	0.77	0.78	0.73	0.76	0.76	0.73	20000
Weighted avg	0.77	0.78	0.73	0.76	0.76	0.73	20000
	Accuracy: 0.78 - ROC AUC: 0.969			Accuracy: 0.76 - ROC AUC: 0.967			

Figure 17: Non-Private and Differentially Private Training of Naive Bayes Algorithm

Logistic Regression

We are using the Logistic Regression classifier from Scikit-Learn and its differentially private counterpart from the Diffprivlib library. The privacy guarantee is relaxed compared to the Naive Bayes experiment by selecting an epsilon parameter of 3.0. Furthermore, we provide the maximum L2 norm of the dataset's rows (data_norm = 7.89) as required by the classifier.

The resulting test set performance metrics are as follows:

	Non-private training			**DP training** (ϵ = 3.0)			
Classification	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Test n
Class 0	0.97	0.95	0.96	0.87	0.90	0.88	1999
Class 1	0.93	0.95	0.94	0.83	0.86	0.84	2006
Class 2	0.97	0.96	0.96	0.95	0.94	0.94	2000
Class 3	0.90	0.91	0.91	0.80	0.80	0.80	2006
Class 4	0.94	0.91	0.92	0.92	0.83	0.87	1999
Class 5	0.94	0.96	0.95	0.92	0.95	0.93	1996
Class 6	0.88	0.89	0.88	0.78	0.87	0.82	1993
Class 7	0.93	0.93	0.93	0.83	0.93	0.88	1996
Class 8	0.97	0.97	0.97	0.96	0.96	0.96	2005
Class 9	0.87	0.88	0.87	0.79	0.61	0.69	2000
Macro avg	0.93	0.93	0.93	0.86	0.86	0.86	20000
Weighted avg	0.93	0.93	0.93	0.86	0.86	0.86	20000
	Accuracy: 0.93 - ROC AUC: 0.983			Accuracy: 0.86 - ROC AUC: 0.975			

Figure 18: Non-Private and Differentially Private Training of Logistic Regression Algorithm

Summary of results

- The non-private version of the Random Forest classifier achieves the maximum accuracy level of 97%.
- There is no substantial difference between the non-private and differentially private version of Naive Bayes. In both cases, accuracy is about 20 percentage points below the optimum.
- The difference between the non-private and differentially private versions of logistic regression results in approx. 7 percentage points.

Assessing the Impact of Dataset Size and Privacy Guarantee on Performance

In many cases, one can improve machine learning performance by adding more data – even though there are diminishing returns. This section investigates how to compensate for the accuracy impact of differential private training by increasing the dataset size.

We show results for different privacy levels by adjusting the parameter epsilon in a range of practical applications. Smaller ε-values are related to higher statistical privacy guarantees.

We use the same 10-class dataset as in the previous examples. The following figure illustrates the resulting impact of dataset size and privacy guarantee on model performance (measured by the auc-score):

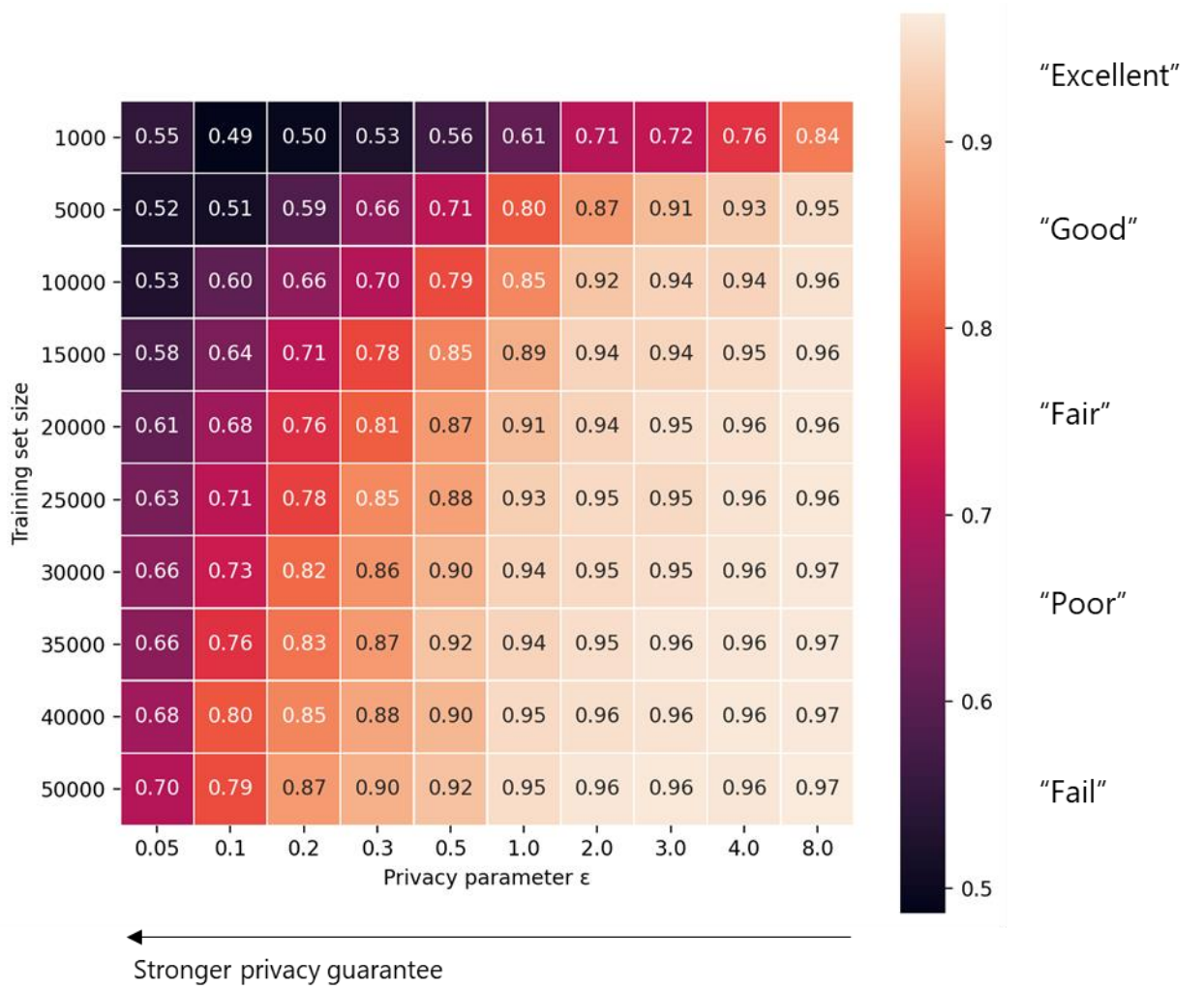


Figure 19: auc-score of Differentially Private Naive Bayes

We have chosen auc-score as a performance metric since it is generally more reliable than the simple accuracy metric. Auc-scores range from 0.5 (quality of a random guess) to 1.0 (perfect classifier). The scores are reported based on the test set. Each cell shows an average metric resulting from 10 experiments (10 classifiers trained on random samples from the training data).

The following table shows commonly accepted guidelines on how different auc-sores relate to model quality:

Auc-Score	Model Quality
0.9 to 1.0	Excellent
0.8 to 0.9	Good
0.7 to 0.8	Fair
0.6 to 0.7	Poor
< 0.6	Fail

Figure 20: Model Assessment Based on auc-score

The scenario of 10,000 training records illustrates the substantial impact of epsilon on the resulting model quality. Ratings range from "failing" at an aggressive privacy level of 0.05 up to "excellent" when epsilon is increased to 2.0 or higher. Also, we notice the clear benefits of adding additional training records even though the effect diminishes quickly for settings with more relaxed privacy levels.

Machine Learning Based on a Differentially Private Dataset

In this section, we introduce another variant of differentially private machine learning. Instead of adding noise during the training process, it is possible to generate a differentially private version of the dataset. This synthetic ("fake") dataset can then be used to train a standard (non-private) machine learning algorithm.

While the synthetic dataset embodies the original data's essential properties, it is mathematically impossible to preserve the full data value and guaranteeing record-level privacy at the same time. Usually, we can't perform arbitrary statistical analysis and machine learning tasks on the synthesized dataset to the same extent as it is possible with the original data. Therefore, the type of downstream task should be considered before the data is synthesized. For a classification problem, data generation can be performed per group of records that belong to the same class. This way, the latent structure of each group is considered when synthetic records are generated. Furthermore, there are challenges in performing machine learning on synthetic datasets with high dimensional and imbalanced datasets.

However, differentially private synthetic data generation is a vibrant field in research with increasing practical applications. New and improved synthesizers are published at a rapid pace. The essential advantage of the synthesizer approach is that the differentially private dataset can be analyzed any number of times without increasing the privacy risk. Therefore, it enables collaboration between several parties, democratizing knowledge, or open dataset initiatives.

We previously used the synthesizer approach to protect sensitive medical information when we compared the privacy protection effectiveness of data anonymization and Differential Privacy in the previous chapters. We had chosen the MWEM synthesizer of SmartNoise for that purpose.

In this case, we choose a more sophisticated approach that improves the utility of synthetic differentially private datasets specifically for machine learning tasks.

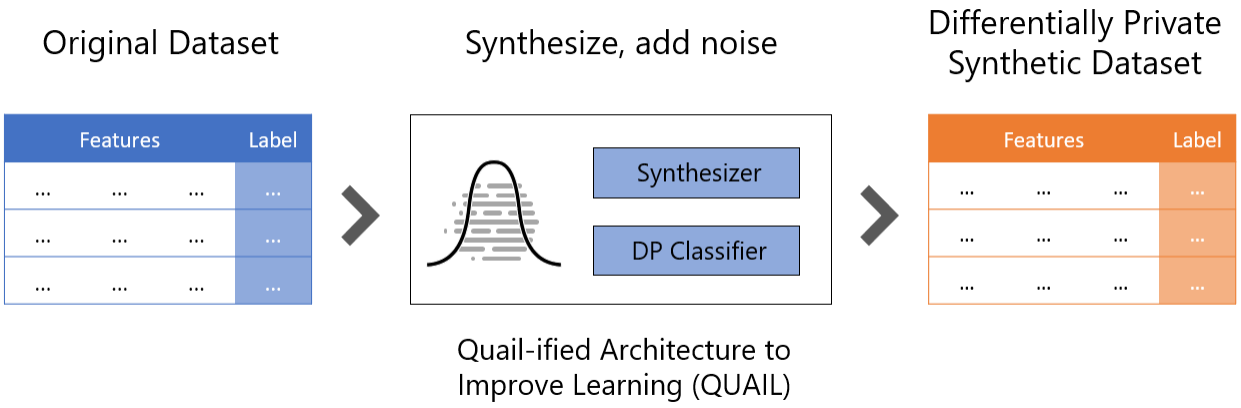
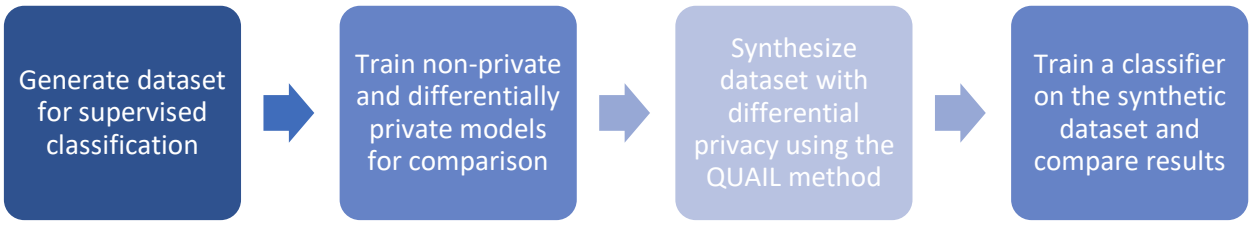


Figure 21: Generating Synthetic Data for Machine Learning Using the QUAIL Method

The Quail-ified Architecture to Improve Learning (QUAIL) is used to generate the differentially private dataset. The method combines a differentially private synthesizer and supervised learning model to produce a synthetic dataset with high utility for machine learning applications.

Our example use case is based on the following workflow:



The demo scenario is available as a Jupyter notebook 5-ml-synthetic-data developed in Python, which you can use to reproduce the results and customize to your use cases.

Like in the previous section, we use Scikit-Learn to generate an artificial multiclass dataset based on the following parameters:

Parameter	Value	Comment
Number of Instances	100,000	80% : 20% Train-test split
Number of classes	10	10 balanced classes
Flipped classes	3%	The fraction of samples which are assigned randomly
Clusters per class	2	2 clusters with a class separator of 5.0
Number of features	7	7 features (float values)

Table 4: Specification of the Artificial Dataset for Differentially Private Machine Learning

Before creating the synthetic dataset, we train two benchmark models for comparison:

		Performance metrics		
Benchmark Model	Overview	Accuracy	F1-score	ROC-AUC
Non-private model	Non-private Random Forest classifier to validate upper bound of achievable accuracy (97% expected based on the dataset)	0.97	0.97	0.984
Differentially private model	A differentially private version of the Logistic Regression algorithm from the diffprivlib library (epsilon of 3.0). See the previous section for details.	0.87	0.87	0.977

Table 5: Benchmark Models

There are several options for choosing the synthesizer and classifier part for QUAIL. For our purposes, we select PATE-CTGAN as a synthesizer and the differential private version of Logistic Regression from the diffprivlib library.

Finally, we train a Random Forest classifier on the resulting synthetic dataset and evaluate its performance on the test set of the original data (train-synthetic-test-real or TSTR).

The following table compares the accuracy metrics of the differentially private Logistic Regression classifier on the original data and the Random Forest classifier trained on the synthetic dataset:

	Differentially private version of Logistic Regression			**Random Forest trained on synthetic data**			
Classification	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Test n
Class 0	0.89	0.90	0.90	0.92	0.94	0.93	2006
Class 1	0.83	0.85	0.84	0.90	0.88	0.89	1998
Class 2	0.95	0.94	0.95	0.97	0.96	0.96	2001
Class 3	0.78	0.81	0.80	0.86	0.88	0.87	1993
Class 4	0.92	0.83	0.87	0.92	0.91	0.92	2000
Class 5	0.94	0.96	0.95	0.96	0.97	0.96	2003
Class 6	0.75	0.87	0.81	0.81	0.92	0.86	1988
Class 7	0.89	0.95	0.92	0.83	0.93	0.88	2006
Class 8	0.97	0.97	0.97	0.97	0.97	0.97	2006
Class 9	0.82	0.64	0.72	0.88	0.64	0.74	1999
Macro avg	0.87	0.87	0.87	0.90	0.90	0.90	20000
Weighted avg	0.87	0.87	0.87	0.90	0.90	0.90	20000
	Accuracy: 0.87 - ROC AUC: 0.977			**Accuracy: 0.90** - ROC AUC: 0.982			

Figure 22: Performance Metrics of Differentially Private Classifier and Machine Learning on Synthetic Data

Machine Learning Results on Synthetic Data

Our example shows that machine learning based on a synthetic dataset can lead to comparable performance as a differentially private classifier. However, research shows that this is not always the case depending on the dataset's characteristics and other factors.

The synthesizer approach's main advantage is that the resulting dataset can be shared and used for analytical purposes any number of times without increasing the risk of privacy loss.

Another advantage is that the synthesizer allows producing an arbitrary amount of data derived from the original dataset's distribution. This could be a promising approach for data augmentation to improve the resulting machine learning model's quality.

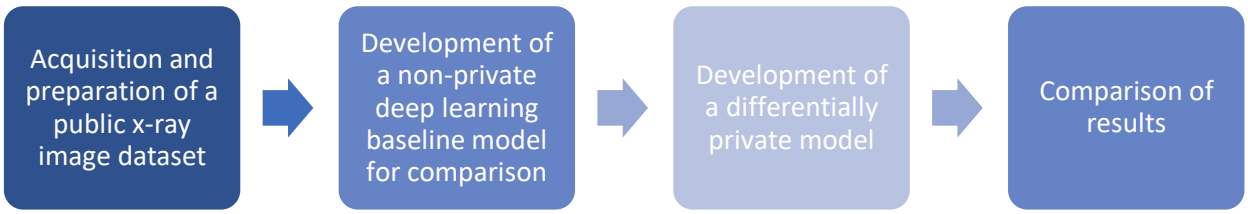
Deep Learning on Medical Images

The previous examples have demonstrated that Differential Privacy can be used to effectively protect private data while enabling analytical use cases and machine learning on structured data with promising results at the same time.

This chapter discusses a more ambitious use case of performing deep learning with Differential Privacy to classify x-ray images to detect pneumonia.

Modern deep learning architectures are especially prone to privacy risks due to their enormous capacity, often with millions (sometimes billions) of trainable parameters. Thus, there is always a tendency to memorize individual data points (or even whole datasets). This behavior is undesired not only for privacy reasons. Machine learning aims to train a model that can learn general patterns to make accurate predictions for new data points instead of memorizing the training data.

The medical imaging example is based on the following workflow:



The corresponding demo scenario is available as a Jupyter notebook [6-deep-learning-medical] developed in Python, which can be used to reproduce the results and to be customized to other use cases. We have selected the deep learning framework PyTorch for our experiment.

Acquisition and Preparation of X-Ray Image Dataset

A relatively small public dataset of medical images has been chosen to keep the scenario straightforward and reproducible with limited computing resources.[4] The dataset contains 5,218 x-ray images with two classes of diagnostic outcomes: 3,876 cases with (viral or bacterial) pneumonia and 1,342 cases without findings ("Normal").

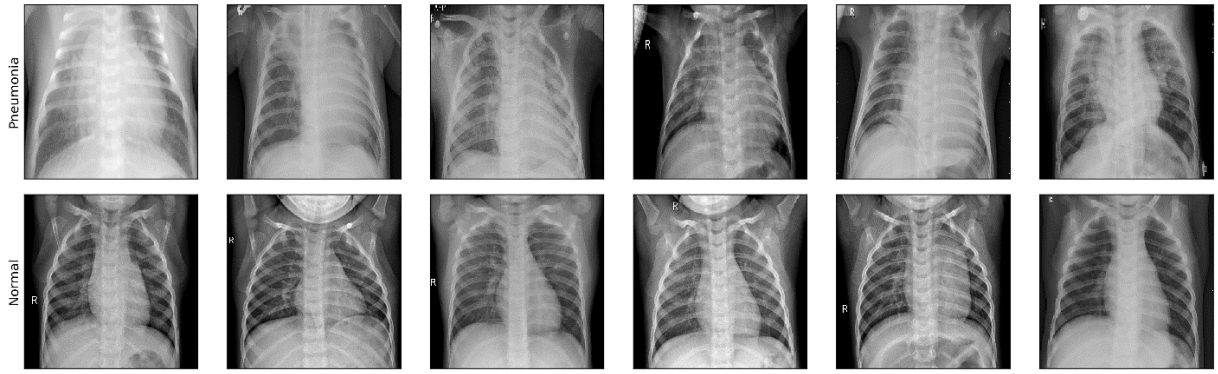


Figure 23: X-Rays Showing "Pneumonia" Versus "Normal" Chest Images

The dataset is split into training and validation sets by a ratio of 90% to 10%. Since some images represent radiographs from the same patient, it has been ensured that there is no overlap of patients between the training and validation sets.

Development of A Non-Private Baseline Model

Before training a deep learning classification model based on the above dataset, we first have to specify a neural network architecture. Convolutional Neural Networks (CNNs or Convnets) represent the state of the art for most computer vision tasks.

There are many architecture options with increasing levels of complexity available. To avoid unnecessary demand for computational resources, long training times, and the risk of overfitting, it is recommended to aim for the simplest model required for the respective data. Since the dataset contains a relatively low number of images, and there are only two classes to distinguish, we have chosen the following architecture:

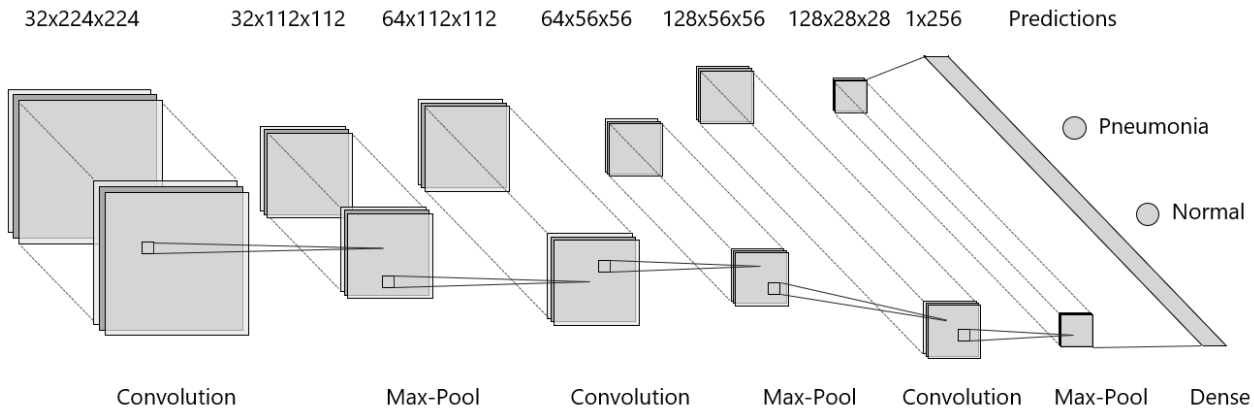


Figure 24: Convolutional Neural Network Architecture for Medical Image Classification

We recommend considering more powerful state of the art architectures like Resnet or Inception for productive medical use cases. Furthermore, strategies like transfer learning (reusing trained networks from comparable computer vision tasks) should be considered. These techniques generally also work with the Differential Privacy approach for deep learning discussed in this paper.

Neural Network Design

Though a detailed discussion of the architecture and functionality of Convnets is outside the scope of this whitepaper, the following summary provides a brief overview of the design:

- The x-ray images are resized to a 224 x 224 pixel resolution before being fed into the Convnet. Other medical imaging use cases will most likely require higher resolutions. However, for the selected dataset, high accuracy results can be achieved with this small image size.
- During the data flow through the Convnet, relevant properties for the classification task (features) are extracted in a hierarchical way. The lower layers of the network detect low-level features like edges or surfaces. More complex features (for detecting pneumonia in this case) are extracted at higher layers. The three convolutional layers perform the detection of features at different abstraction levels in the network, where the images are scanned by a small moving window (kernel).
- To reduce computational effort while focusing on the most dominant features, the image size is reduced further as the data flows through the three max pooling layers.
- Two dropout layers are included to reduce the risk of overfitting to the training data.
- The final layer consists of two neurons for representing the classes "pneumonia" and "normal." A multiclass architecture has been selected since one of the evaluated Differential Privacy modules was not yet compatible with binary classification.

The model is trained with a straightforward training approach using PyTorch as the deep learning framework. No data augmentation (generating artificial samples by transforming training images) or transfer learning has been used. The images have been split into a training and validation set to identify potential overfitting during training. Adam was used as the optimizer with a learning rate of 0.0001.

The model achieves high accuracy values already after the images have passed the training process for 10 times (10 epochs with a batch size of 32):

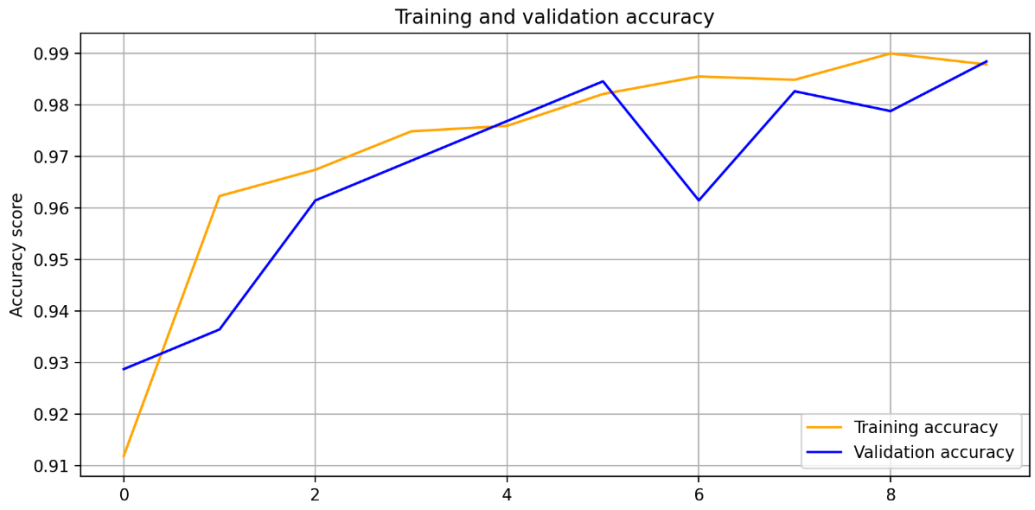


Figure 25: Progress of Typical (Non-Private) Training Process for Image Classification

The training was performed on an Azure Standard NC 12 Tesla K80 GPU Compute Instance with 112 GB memory (GPU memory of 24 GB). The training process took 19 minutes.

The following table summarizes the model performance based on validation set results:

Accuracy score: 0.981 auc-score: 0.999				
Classification report	Precision	Recall	F1-Score	Support
Normal	1.00	0.93	0.96	134
Pneumonia	0.97	1.00	0.99	385
Micro avg	0.98	0.98	0.98	519
Macro avg	0.99	0.96	0.97	519
Weighted avg	0.98	0.98	0.98	519
Samples avg	0.98	0.98	0.98	519

Table 6: Validation Set Classification Performance of Non-Private Training

The validation results indicate a strong classification performance. 98% of the validation images are classified correctly. A good practice is to compare these metrics with a simple baseline approach. Since 74% of validation images belong to the pneumonia class, a trivial (nonsense) classifier that predicts all cases as "pneumonia" would achieve 74% accuracy. The performance achieved with the trained model is 24% percentage points above the trivial classifier. The classification report provides additional insight into its performance: The recall scores show that 93% of actual normal cases are correctly identified as such. Furthermore, 100% of real pneumonia cases in the validation set were correctly recognized.

For a real-life use case, further evaluation would be required. Production models are generally evaluated on an additional test set apart from the training and validation sets. Using just the validation set is considered sufficient to compare non-private and differentially private training approaches in this paper.

Development of A Differentially Private Model

The identical dataset and neural network architecture have been used to ensure an objective comparison between the non-private and differentially private model. The difference between both approaches is the way how the differentially private model is trained.

Differential Privacy is not included in the PyTorch deep learning library by default. Therefore, a compatible Differential Privacy library must be used. We have chosen PyTorch-Opacus (formerly known as PyTorch-DP). There are additional Differential Privacy libraries available for PyTorch and for the TensorFlow deep learning library (e.g., TensorFlow Privacy).

The key aspect of Differential Privacy in deep learning used in this paper is to add carefully tuned statistical noise to the neural network's training process to mask the contribution of individual data points (patient x-ray images in this case). Therefore, the risk that inferences could be drawn on individuals' data points from the trained model is reduced.

Since the training process is vital for a better understanding of this approach, the following description aims to provide a high-level intuition about how deep neural networks learn from training data.

Before starting the training process, the following artifacts exist:

- The training images, each labeled by radiologists with the actual diagnostic outcome ("normal" or "pneumonia")
- An (untrained) network structure that takes x-ray images as input, processes them layer by layer to finally predict the class. The network consists of many parameters (called weights and biases), determining how the images are processed as they move through the network.

These parameters are initialized with random numbers before the training process. Therefore, any predictions of the neural network are "random guesses" at that time.

The main element of training is an optimization process where these parameters are iteratively adjusted so that the network's predictions better align with the known correct labels of the training data (see the description of the Stochastic Gradient Descent Algorithm).

In more technical terms, an error function (also called loss function), which measures the mismatch between the actual labels and the network's predictions, is minimized. Due to the random starting position, the network makes many wrong predictions initially. Therefore, the error is high at the beginning. The error value is gradually reduced during training as the model makes better predictions by learning from the training data.

Stochastic Gradient Descent Algorithm

The following illustration describes the iterative learning process for two parameters $weight_1$ and $weight_2$ and the corresponding error function:

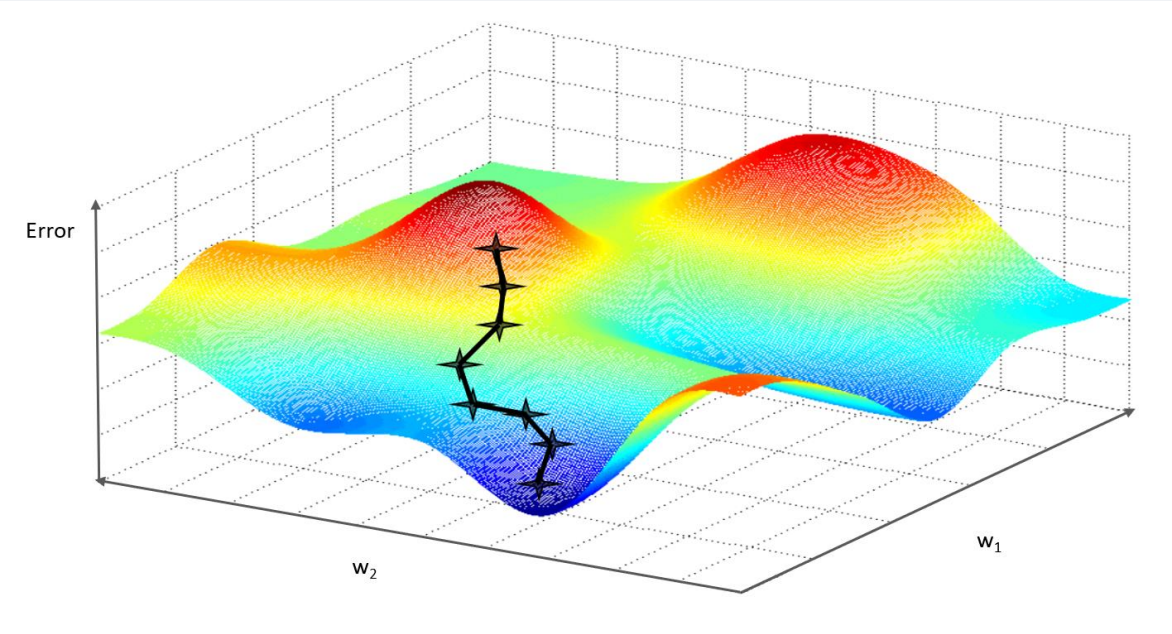


Figure 26: Stochastic Gradient Descent (SGD) Algorithm for Two Parameters

A common allegory for this optimization process is to imagine the error function being a mountain and a hiker standing at the peak, who wants to reach the valley as quickly as possible. Since the mountaineer is surrounded by heavy fog, she only sees her immediate surroundings but cannot plan her path downwards further ahead at any given location.

In neural network training, the method of iteratively finding the path to the valley of the "error mountain" is called gradient descent and roughly works as follows:

1. Make predictions for a randomly sampled subset (called minibatch) of the data based on the available knowledge (namely using the current parameters $weight_1$ and $weight_2$).
2. Calculate the error by comparing the predictions with the known true outcomes (labels) from the training data.
3. Find the direction of the steepest downward slope from the perspective of the current position. This is done by calculating the gradient (the derivative of the loss function with respect to the parameters). The gradient acts as a signpost to the direction in which the error function is decreased by the most amount. This approach corresponds to the hiker's scenario, who chooses the direction of steepest descent within her actual limited visual range.
4. Take a few steps in the direction of the steepest downward slope.

This procedure is repeated until the valley is reached, respectively, the improvement of the error value diminishes.

The first diagram in Figure 25 shows this optimization procedure in action. The error value improves as the network updates its parameters in 1,470 iterations (10 epochs, each with 147 updates since the images are processed in batches of 32).

The goal of using Differential Privacy is to limit the risk of exposing sensitive data from the model's training set by providing a provable (statistical) privacy guarantee. In particular, the trained machine learning model should not be affected by individual data points from the training images.

In deep learning, Differential Privacy can be implemented by making the following adjustments to the third step in the outlined Stochastic Gradient Descent (SGD) algorithm:

- Bounding the gradient to limit its sensitivity to individual data points
- Adding random noise to the gradient calculation

These adjustments are illustrated in the following figure:

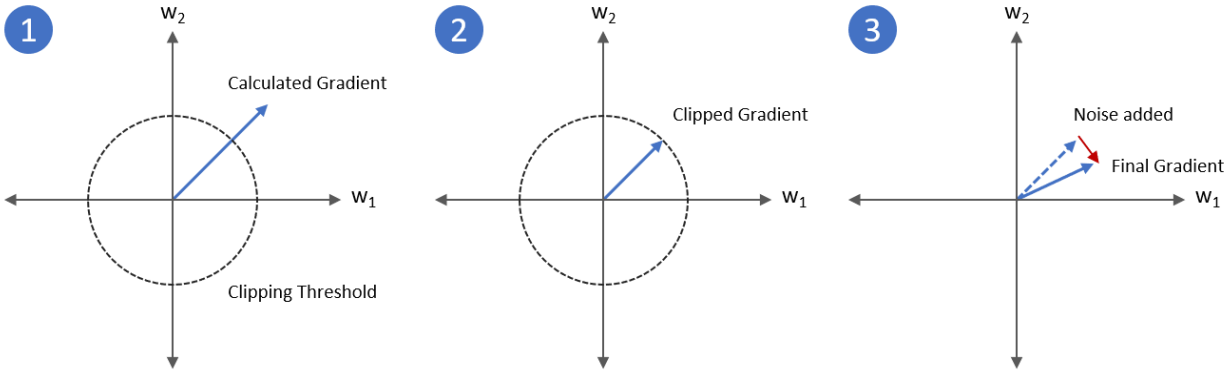


Figure 27: Differentially Private Adjustments to the Gradient Descent Algorithm[5]

1. The gradient is calculated according to the usual SGD-algorithm (in a non-private fashion)
2. The gradient's sensitivity to individual data points is bound (also known as gradient clipping). Consequently, the extent to which each data point can impact the model's parameters is also limited. The gradient clipping amount is a hyperparameter (called `max_grad_norm` in Opacus) that is defined before training the model. For the medical imaging demo case, we have chosen to set `max_grad_norm = 2`.
3. To prevent the identification (if a particular data point is included in the training set) a defined amount of noise is added to the clipped gradient. The amount of noise is also a hyperparameter (called `noise_multiplier` in Opacus). Higher values increase the privacy guarantee but make the training process more difficult and thus negatively impact the model's accuracy. For the medical imaging demo, we have chosen a `noise_multiplier` of 0.65.

Integrating Opacus into Machine Learning Workflows

Opacus (formally known as PyTorch-DP) makes it easy to integrate differential privacy into an existing ML pipeline:

```
from opacus import PrivacyEngine
model = Cnn().to(device)

# DP specific code:
optimizer = torch.optim.Adam(model.parameters(), lr = 0.0007)
privacy_engine = PrivacyEngine(
    model,
    batch_size = 32,
    sample_size = len(training_loader.dataset),
    alphas = [1 + x / 10.0 for x in range(1, 100)] + list(range(12, 64))
    noise_multiplier = 0.65,
    max_grad_norm = 2.0)
privacy_engine.attach(optimizer)

# Continue by training the model as usual
history = train(model = model,
                optimizer = optimizer,
                loss_fn = nn.CrossEntropyLoss(),
                train_dl = training_loader,
                val_dl = validation_loader,
                epochs = 25)
```

Besides importing PrivacyEngine, only the framed code is specific to Differential Privacy training. The model is instantiated in the same way as in the non-private case. The same applies to the training procedure. A custom function is used, which does not need to be adjusted for achieving Differential Privacy (except for optionally reporting privacy parameters after each epoch). Refer to the collateral Jupyter notebooks for further details.

The training was performed on the same Azure Standard NC 12 GPU Compute Instance. The following illustration shows the training progress over 25 epochs:

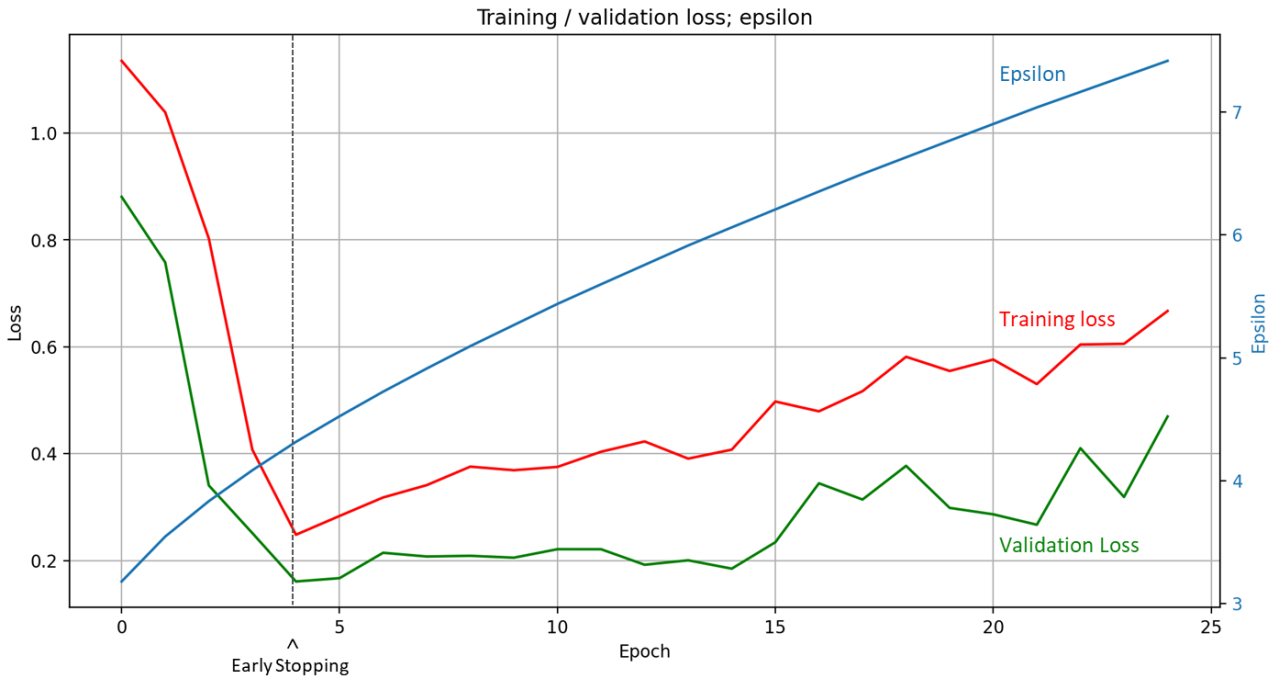


Figure 28: Progress of Error and Privacy Guarantee during Differentially Private Training

In this example, ending the training already after the fourth epoch makes sense since it is associated with the minimum training and validation error (and, therefore, maximum accuracy). Furthermore, additional training decreases the privacy guarantee (reflected by an increasing ε value) due to the repeated loops through the sensitive training data.

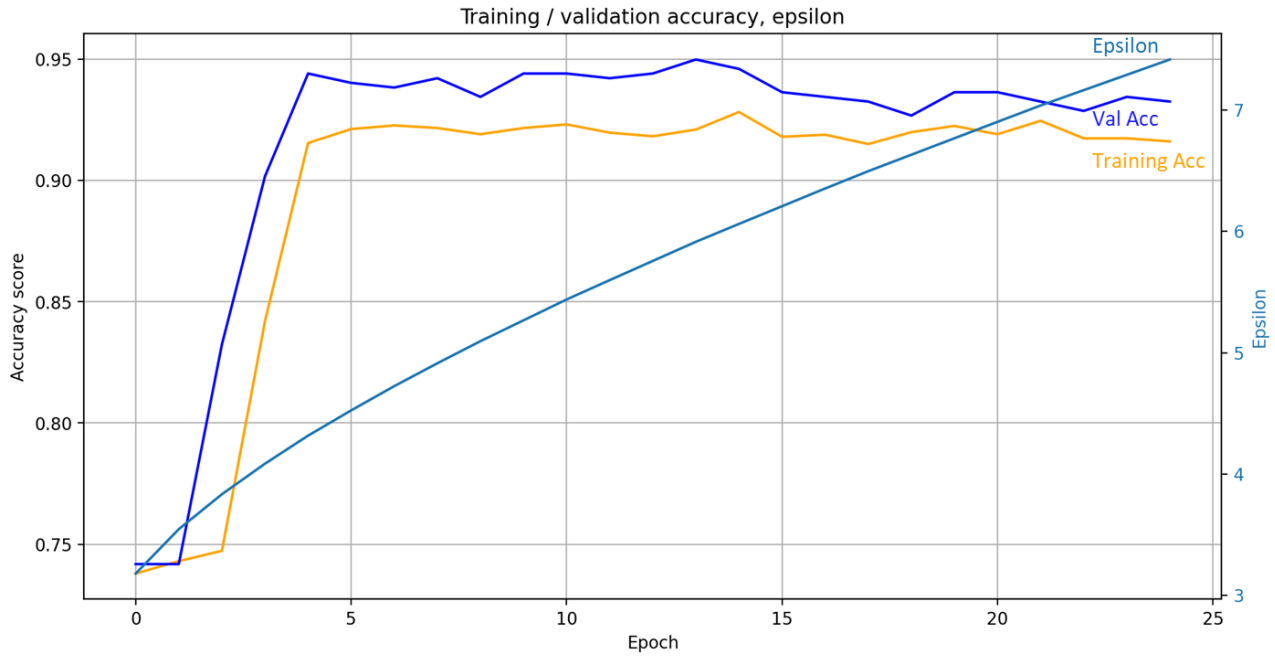


Figure 29: Progress of Accuracy Score and Privacy Guarantee during Differentially Private Training

The following table summarizes the model performance based on validation set results:

Accuracy score: 0.944 auc-score: 0.979				
Classification report	Precision	Recall	F1-Score	Support
Normal	0.92	0.86	0.89	134
Pneumonia	0.95	1.97	0.96	385
Micro avg	0.94	0.94	0.94	519
Macro avg	0.94	0.92	0.93	519
Weighted avg	0.94	0.94	0.94	519
Samples avg	0.94	0.94	0.94	519

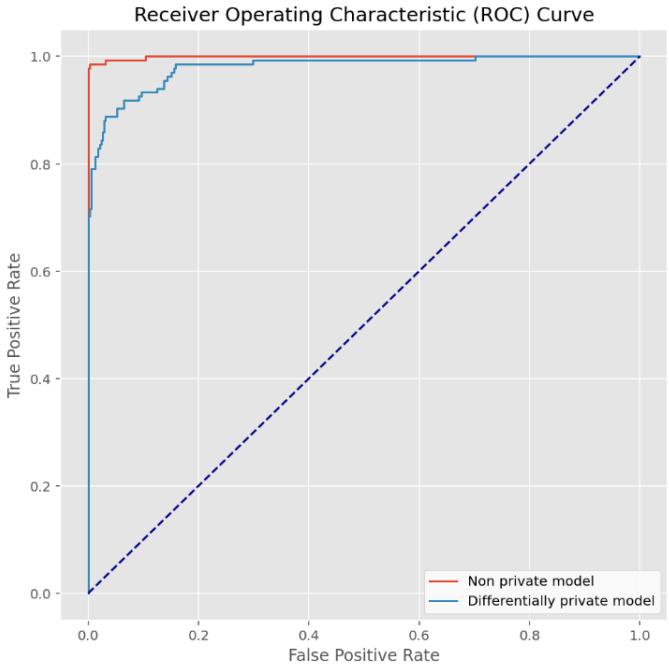
Table 7: Validation Set Classification Performance of Differentially Private Training

The diagram shows what we have also observed in other Differential Privacy training settings: The initial loss is significantly higher than in non-private training, and accuracy tends to stay constant for a few epochs. We have observed this behavior with other configurations for up to 8 epochs.

Comparison of Results

The following statistics provide an overview of the performance of both models in comparison:

Metric	Non-priv. model	DP model
Accuracy Score	0.981	0.944
Auc-score	0.999	0.979
F1-Score Normal	0.96	0.89
F1-Score Pneumonia	0.99	0.96
Weighted Avg	0.98	0.94
Privacy Guarantee ε	N/A	4.32
Training Time / epoch	116 s	145 s



The accuracy of the differentially private model is 3.7 percentage points below the performance of the non-private model. The other metrics show a similar order of magnitude. The f1-scores show a more significant difference in the ability to detect the underrepresented "normal" class.

Choosing the Privacy Parameter Epsilon

Generally accepted guidelines about desirable or acceptable values of the privacy parameter epsilon have yet to be established.

- Lower values are associated with stronger privacy guarantees and hence desirable from the data protection perspective. From the model quality point of view, too aggressive configurations (low values of epsilon) should be avoided.
- Considering the business use case and the amount of risk that we are willing to accept helps to make the privacy parameter tangible. It is possible to estimate the risk in monetary terms in some financial settings (e.g., the maximum expected amount of increase in insurance premium).
- Researchers from the Harvard University provide preliminary guidance: "As a rule of thumb, however, ε should be thought of as a small number, between approximately 1/1000 and 1."[6]
- In practical applications, strong protection can often be achieved with higher privacy budgets. Epsilon is an upper bound measure of the risk and therefore describes a worst-case scenario. Primarily in use cases of deep learning, research has shown that higher values are often acceptable. Some experiments showed that no leakage of individual data points from trained deep learning models was possible even at epsilon levels of 50.[7]

There are several measures which help improve the model performance further:

Increasing the Size of the Dataset

The medial dataset was chosen for simplicity and the ability to reproduce results with limited computing resources. However, for deep learning standards, 5,000 images are making a relatively small dataset. The following example of a differential private training with the MNIST dataset of handwritten digits illustrates the benefits of increasing the training set size:

Even though there are apparent diminishing returns, the validation accuracy is increased to 95% by adjusting the training set size up to 60,000 images. Simultaneously, the privacy guarantee can be improved, which is reflected by the reduction of the privacy parameter epsilon.

Reducing the Model Size

With differentially private training, there is a trade-off between model size and accuracy. A best practice recommendation is to reduce the number of parameters since this makes it easier to get good results with Differential Privacy. Besides making reductions to the network architecture, also transfer learning might be used to achieve this goal as it can reduce the number of parameters that need to be trained.

Transfer Learning

In the medical imaging example, the model was initialized with random parameters (trained from scratch). Transfer learning strategies reuse parts of an already trained model. This approach typically reduces training time and can also improve results. A Differential Privacy use case could use an existing pre-trained model (e.g., from a public x-ray dataset) and perform a differentially private training using the private dataset on top of the pre-trained model.

The following illustration provides a conceptual overview of transfer learning.

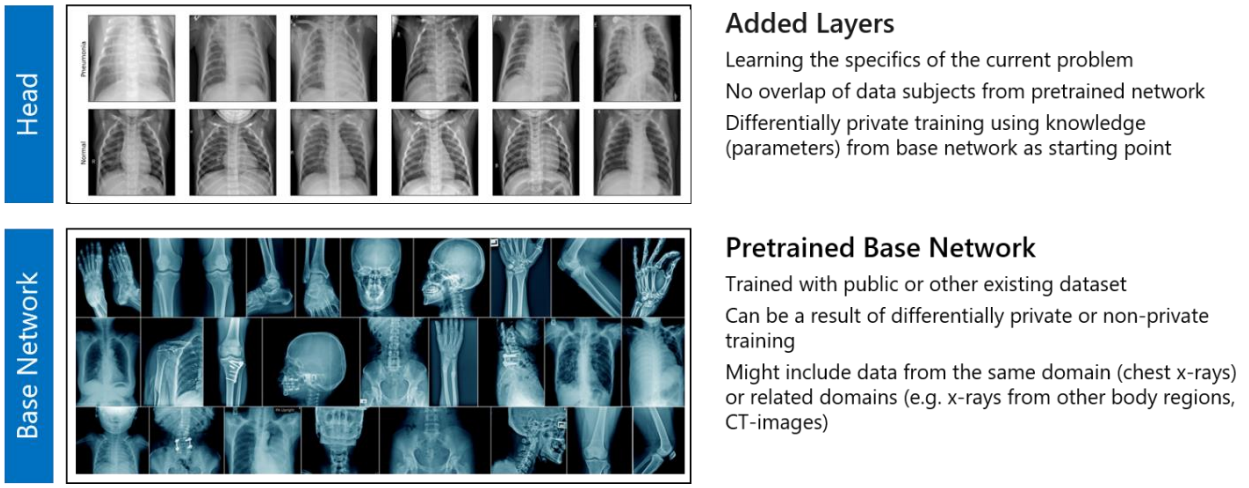


Figure 30: Transfer Learning on Medical Images

The resulting model likely benefits from transfer learning even if the base network was trained on images from another (but potentially related) computer vision use case. Even in less related use cases, the network is likely able to reuse trained knowledge to detect basic visual concepts (e.g., edges, patterns, surfaces).

Hyperparameter Tuning

Hyperparameter tuning typically includes trying alternative architectures, model- or training-related parameters (e.g., activation functions, learning rate, batch size, optimizers). Furthermore, the parameters specific to Differential Privacy (e.g., noise multiplier, gradient clipping, alpha) significantly impact the model's performance. Depending on the privacy requirements, these parameters could be further relaxed to improve the final model's performance.

Conclusion

The increasing number of severe breaches of sensitive personal data shows that today's disclosure limitation practices are often insufficient. Popular data anonymization techniques are vulnerable to privacy attacks like re-identifying individuals in sensitive datasets.

Differential Privacy is the emerging gold standard of privacy protection. It protects user privacy by adding carefully tuned random noise to data and computations.

Practical applications range from primary statistical and analytical use cases to advanced scenarios like machine learning and deep learning. We have seen that the introduction of noise leads to a trade-off between privacy protection and analytical results accuracy. However, one can often compensate for the loss of data utility by adding more data.

Communication of the concept and its benefits to various stakeholders like data subjects, analytics practitioners, regulators, and decision makers is critical. Individuals might show a higher willingness to provide personal data if a formal privacy guarantee is given and proven to hold.

The SmartNoise system is an open source implementation based on the Differential Privacy concept. It is developed by Microsoft, Harvard's Institute for Quantitative Social Science, and the School of Engineering and Applied Sciences as part of the Open Differential Privacy (OpenDP) initiative. The project aims to connect solutions from the research community with the lessons learned from real-world deployments to make Differential Privacy broadly accessible.

A vital goal of this paper is to introduce Differential Privacy to practitioners and decision makers. We hope that the interactive demo scenarios help you experience the concept in action and apply them to your use cases.

References

For information on the topics presented here, please note the links and sources below.

---

1 Sweeney, Latanya. "Simple demographics often identify people uniquely." Health (San Francisco) 671.2000 (2000): 1-34.

2 Narayanan, Arvind, and Vitaly Shmatikov. "Robust de-anonymization of large sparse datasets." 2008 IEEE Symposium on Security and Privacy (sp 2008). IEEE, 2008.

3 https://openreview.net/pdf?id=ABZSAe9gNeg

4 https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia (Dataset source)

5 Based on the illustration by Neeraj Rajkumar Parmaar (Differential Privacy in Deep Learning, https://towardsdatascience.com/differential-privacy-in-deep-learning-cf9cc3591d28)

6 Wood, Alexandra, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, James Honaker, Kobbi Nissim, David R. OBrien, Thomas Steinke, and Salil Vadhan. “Differential privacy: A primer for a non-technical audience.” Vanderbilt Journal of Entertainment & Technology Law 21, no. 1 (2018): 209-275.

7 TensorFlow Privacy: Learning with differential privacy for training data - Ulfar Erlingsson (Google Brain)