UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

The Curse of Dimensionality for Additive Noise

Der Fluch der Dimensionalität bei additivem Noise

Masterarbeit

im Rahmen des Studiengangs
Informatik
der Universität zu Lübeck

vorgelegt von
Marven Kummerfeldt

ausgegeben und betreut von
Prof. Dr. Esfandiar Mohammadi

Lübeck, den 11. Juli 2021

Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

____

Lübeck, 11. Juli 2021

Abstract

A common method to protect personal information in data-dependent algorithms while adding provable privacy guarantees is Differential Privacy (DP). To achieve this, DP generally uses a two-stage approach: First, it quantifies the maximum impact that a single data point can have on the output of an algorithm. Then, noise is scaled according to this maximum impact and added to the output of the algorithm to hide said influence. This approach more often than not results in an immense amount of added noise and thus a significant decrease in utility when compared to not-privacy-preserving algorithms.
The initial goal of this thesis is to address the described decrease in utility and to investigate whether an additional voting step can be used to reduce the negative impact of additive noise on the performance of a differentially private algorithm. This investigation reveals an inherent problem of additive noise, the analysis of which constitutes the main part of this thesis. Our main result shows that the negative impact of additive noise increases with the dimensionality of the output to be perturbed, regardless of the noise distribution used. We think that these findings can steer future research on differentially private mechanisms on high-dimensional outputs.

Zusammenfassung

Eine bewährte Methode, persönliche Informationen in datenabhängigen Algorithmen zu schützen und gleichzeitig beweisbare Datenschutzgarantien zu erreichen, ist Differential Privacy (DP). Hierfür verwendet DP im Allgemeinen einen zweistufigen Ansatz: Zunächst wird die maximale Auswirkung quantifiziert, die ein einzelner Datenpunkt auf das Ergebnis eines Algorithmus haben kann. Danach wird Noise entsprechend dieser maximalen Auswirkung skaliert und zum Ergebnis des Algorithmus hinzugefügt, um den beschriebenen Einfluss zu verbergen. Dieser Ansatz führt in den meisten Fällen zu einer erheblichen Menge an additivem Noise und damit zu einer deutlichen Verringerung der Ergebnisqualität im Vergleich zu nicht die Privatsphäre schützenden Algorithmen.
Das initiale Ziel dieser Arbeit ist es, die beschriebene Verringerung der Ergebnisqualität zu adressieren und zu untersuchen, inwiefern ein zusätzlicher Abstimmungsschritt verwendet werden kann, um den negativen Einfluss von additivem Noise auf die Leistung eines die Privatsphäre schützenden Algorithmus zu reduzieren. Diese Untersuchung offenbart ein inhärentes Problem von additivem Noise, dessen Analyse den Hauptteil dieser Arbeit darstellt. Unser Hauptergebnis zeigt, dass der negative Einfluss von additivem Noise mit der Dimensionalität der zu schützenden Ausgabe zunimmt, unabhängig von der verwendeten Noise-Verteilung. Wir denken, dass diese Erkenntnisse die zukünftige Forschung zu die Privatsphäre schützenden Mechanismen auf hochdimensionalen Ausgaben lenken können.

Contents

List of Figures

List of Algorithms

1 Introduction

With the rapid technological developments of the last decade, data-dependent algorithms became interesting to a variety of applications. Whether it's traffic control, product recommendation, speech recognition or even medical applications, especially with the advent of Machine Learning (ML), the increasing reliance on immense amounts of data has reached large parts of society.
Out of this development, an inherent conflict became increasingly relevant: On the one hand, data-dependent algorithms can be used to benefit people, companies and entire societies. The more data is available and the more detailed this data is, the better these algorithms generally perform. On the other hand, there is a growing and justified concern about the automated analysis and processing of personal data by companies or governments.
These concerns have long since reached the public, as evidenced by both corporate statements (e.g., privacy segment at this year's Apple WWDC21 [App21]) and legislative resolutions (e.g., the *General Data Protection Regulation* (GDPR) in the European Union [Eur16]). Nevertheless, the seemingly irresolvable contradiction described above between an extensive use of personal information and data minimization remains.
This motivated a variety of research in the field of privacy-preservation with the goal of integrating data protection and provable privacy guarantees into data-dependent algorithms. Famous representatives of this research are *k-anonymity* by [SS98] and *Differential Privacy* (DP) by [DMNS06]. Both techniques allow algorithms and their users to learn and use information about groups in a dataset, but protect information about individuals. Unfortunately, previous results show that even with these sophisticated methods, privacy-preservation always comes at the expense of utility.
The thesis at hand focuses on a common problem that many approaches in DP suffer from. To achieve privacy, these approaches use additive noise to hide the impact that individuals have on the outcome of an algorithm. This method often adds a lot of noise, causing the aforementioned decrease in utility. The first part of this thesis investigates whether an additional voting step could be used to limit the negative effects of additive noise on differentially-private algorithms. Then the second part further analyses general properties of additive noise and shows why additive noise inevitably leads to a decrease in utility. Further, it is evinced that this adverse effect increases with the dimensionality of the noise being added. As a result, it is shown that especially for high-dimensional

applications, such as those found in ML, additive noise is unable to preserve much of an algorithm's utility, which should motivate further research to find better alternatives to achieve DP in high-dimensional applications.

1.1 Structure of the Thesis

The structure of this thesis is as follows: At the beginning, Chapter 2 describes the theoretical foundations of DP and shows how it is currently applied to the training of deep neural networks in an algorithm called DP-SGD. Afterward, Chapter 3 describes and analyses the idea of adding an additional voting step to a differentially-private algorithm. As an exemplary application, DP-SGD is chosen. Chapter 4 is then dedicated to the analysis and implications of additive noise in DP. Finally, Chapters 5 and 6 conclude this thesis and motivate possible future work.

2 Preliminaries

2.1 Differential Privacy

A common method to incorporate privacy-preserving techniques into data-dependent algorithms while at the same time providing provable privacy guarantees is *differential privacy* (DP). The definition of DP dates back to 2006 where it was first proposed by [DMNS06] after being preceded by prior work e.g., [DN03, DN04, BDMN05]. The original goal of DP is to provide privacy guarantees for databases and their querying while preserving the utility of such queries. The observations made through query outputs should hide whether a single entry is part of a database or not. An intuitive presentation of DP as a cryptographic puzzle is shown in Figure 2.1: An attacker sends two databases, D_0 and D_1, to a challenger which can be understood as a *cryptographic oracle*. The challenger then randomly chooses one of the databases, applies the mechanism $\mathcal{M}$ to this database and sends the result back to the attacker. From this result, the attacker's chance of telling whether D_0 or D_1 was used in $\mathcal{M}$ to produce the result should be bounded by $\exp(\varepsilon)$. With regards to the relationship of D_0 and D_1, DP only restricts them to be neighboring or *adjacent*.

Definition 2.1 (Adjacent Datasets). Two datasets D_0 and D_1 are called adjacent or neighboring if they only differ in a single element $\{x\} = (D_0 \setminus D_1) \cup (D_1 \setminus D_0)$. x is often called the *challenge element*.

With this definition, DP is formally defined as:

Definition 2.2 (Differential Privacy (DP)). A randomized mechanism $M : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ is ε-differentially private if for any two adjacent datasets $D_0, D_1 \in \mathcal{D}$, all x and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ the following holds:

$$\exp(-\varepsilon) \le \frac{\Pr[M(D_1) \in \mathcal{S}]}{\Pr[M(D_0) \in \mathcal{S}]} = \frac{\Pr[M(D_0 \cup \{x\}) \in \mathcal{S}]}{\Pr[M(D_0) \in \mathcal{S}]} \le \exp(\varepsilon)$$

ε is the main parameter which quantifies the privacy guarantees of a mechanism. The lower the ε, the higher the degree of privacy protection which mechanism $\mathcal{M}$ achieves. In practice, an $\varepsilon \le 1$ has shown to provide strong privacy guarantees, whereas the guarantees of an $1 < \varepsilon \le 10$ are considered weak and with any $\varepsilon > 10$ the degree of privacy

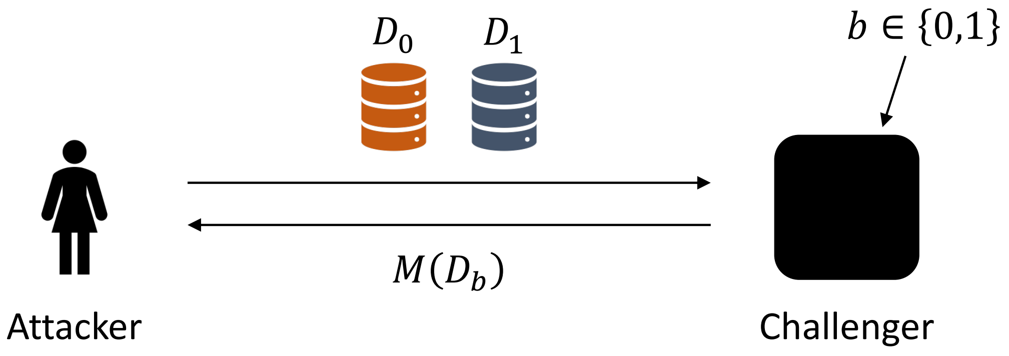


Figure 2.1: Differential privacy as a cryptographic puzzle.

protection is negligible.

As seen in Definition 2.2 differential privacy not only defines privacy guarantees for databases and their querying but for arbitrary mechanisms and thereby arbitrary applications. For example, when talking about machine learning, DP can be used to apply privacy guarantees to the training of a neural network and quantify the probability of an attacker being able to tell whether a certain training sample was used during training (so-called membership inference attack [SSSS17]) or even extract training data from a trained model (so-called model inversion attack [FJR15]).
The main method to achieve DP is the *sensitivity method*, also known as *additive mechanism*, as proposed by [DMNS06]. Its goal is to quantify the maximum influence a single input sample can have on the output of a mechanism and add noise appropriately scaled to cover this influence. The magnitude by which a single input sample can change a function or mechanism f is called *sensitivity* of f.

Definition 2.3 (Sensitivity). The magnitude by which a single input sample can change the function f in the worst case is called *sensitivity* of f or S_f.

Since the sensitivity of a mechanism is often arbitrary, it has to be bounded artificially to represent an appropriate scale for the noise required to achieve DP. This technique is called *clipping*. As an example, one could consider a mechanism $\mathcal{M}_{sum}(a, b) = a + b$ which calculates the sum of two positive integers a and b. It is easy to see that the sensitivity of $\mathcal{M}_{sum}$ is unbounded since a and b can be of arbitrary positive values. When applying clipping, a clipping bound C is introduced. C limits the influence of a and b respectively on $\mathcal{M}_{sum}$ by fixing their maximum value. Any $a, b > C$ is simply set to C, whereas values $\leq C$ remain as is. As a consequence, the maximum impact a, b can have on $\mathcal{M}_{sum}$ is bounded by C and thus $S_{\mathcal{M}_{sum}} = C$.

In the years since the introduction of DP it became apparent that the bounds suggested by Definition 2.2 are too strict in most scenarios. E.g., when analysing the privacy guarantees of the *Gaussian distribution* using Definition 2.2 it does not suffice to achieve ε-DP. That is, although only resulting from the characteristics of the tails of the Gaussian's probability density function (PDF) and thus only from a very small probability mass which is unlikely to play a role in practice.
To allow for a small probability of the privacy guarantees of a mechanism to fail, *approximate differential privacy* was introduced.

Definition 2.4 (Approximate Differential Privacy (ADP)). A randomized mechanism $M : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ is called (ε, δ)-differentially private if for any two adjacent datasets $D_0, D_1 \in \mathcal{D}$ and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ the following holds:

$$\Pr[M(D_0) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[M(D_1) \in \mathcal{S}] + \delta$$

The additional parameter δ describes the probability of the privacy guarantees of a mechanism to fail completely. In other words, it describes the probability of events to occur that clearly distinct one input data from another. Such events are called *distinguishing events*. If a mechanism $\mathcal{M}$ has no distinguishing events, *M* is called *purely differentially private*.

Definition 2.5 (Pure Differential Privacy (PDP)). A randomized mechanism $M : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ is called purely-differentially private if for any two adjacent datasets $D_0, D_1 \in \mathcal{D}$ and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ the mechanism is $(\varepsilon, \delta = 0)$-differentially private.

Currently, a large share of research in DP focuses on finding and proving tight (ε, δ)-bounds for privacy preserving mechanisms. In most cases this is done via the *privacy loss*

$$\mathcal{L}^{(o)}_{\mathcal{M}(x)\|\mathcal{M}(y)} = \ln\left(\frac{\Pr[\mathcal{M}(x) = o]}{\Pr[\mathcal{M}(y) = o]}\right)$$

which compares the probability of a certain observation *o* to occur for a mechanism $\mathcal{M}$ for adjacent inputs x and y. To achieve (ε, δ)-DP, the privacy loss of a mechanism has to be bounded by ε with probability at least $1 - \delta$.
From the definition of the privacy loss, many approaches arose to estimate the true privacy loss of a mechanism more accurately. Amongst the most popular are *concentrated differential privacy* by [DR16], *Rényi differential privacy* by [Mir17], *privacy profiles* by [BBG20] as well as *privacy buckets* by [MM18, SMM19], each having their own advantages and disadvantages.
Another main part of research in DP is the development of differentially private algorithms that achieve DP while preserving as much as possible of the utility of their not-

privacy-preserving counterparts. There are three main techniques based on the mentioned sensitivity method to add DP to an algorithm:

Input perturbation Where the input of an algorithm is perturbed. E.g., this is used by [KLN$^+$20] for *differentially private empirical risk minimization* (DP-ERM).

Objective perturbation Where the objective function of a (learning) algorithm is perturbed. A well-known example is objective perturbation as proposed by [CMS11] that also use it to improve DP-ERM.

Output perturbation Where the output of an algorithm is perturbed. An example is the already mentioned sensitivity method of [DMNS06].

This thesis focuses on the third technique: *output perturbation*. The de-facto standard to use SGD in a privacy-preserving manner is DP-SGD which uses the sensitivity method of [DMNS06] and is discussed in the next section.

2.2 DP-SGD: Applying Differential Privacy to Machine Learning

A common, de-facto standard method to train deep neural networks (DNN) in a privacy-preserving manner is *Differentially Private Stochastic Gradient Descent* (DP-SGD) as proposed by [ACG$^+$16]. DP-SGD adapts the widely used machine learning algorithm *Stochastic Gradient Descent* (SGD) and applies output perturbation to achieve DP. Although suffering from various problems (e.g., an disparate impact on underrepresented classes [XDW20]) it is still a common method to achieve DP when training neural networks.
As for its not-privacy-preserving counterpart SGD, the goal of DP-SGD is to train a machine learning model with parameters θ by minimizing the empirical loss function $\mathcal{L}(\theta)$ over the course of several training steps. To prepare the input data $\{x_1, \ldots, x_N\}$ of size N for the training process, it is split into disjoint subsets, so-called *batches* or *lots*, of size L. Each of these batches is then used to calculate one single training step. This way, an epoch of SGD or DP-SGD training consists of N/L training steps. Since the training is not only performed for a single but for an arbitrary number of epochs, the complete training process consists of $T = N/L \cdot \#epochs$ training steps.
In each training step, the gradients $\nabla_\theta \mathcal{L}(\theta, x_i)$ for all x_i in the current batch are calculated. Afterward, SGD calculates the average gradient of the batch and updates θ in the opposite direction of this average gradient. To achieve DP, DP-SGD alters this update step. According to the sensitivity method of [DMNS06], it first limits the sensitivity of each update step by clipping the length, or more precisely the l_2-norm, of each gradient by a clipping bound C. This sensitivity is called l_2-Sensitivity.

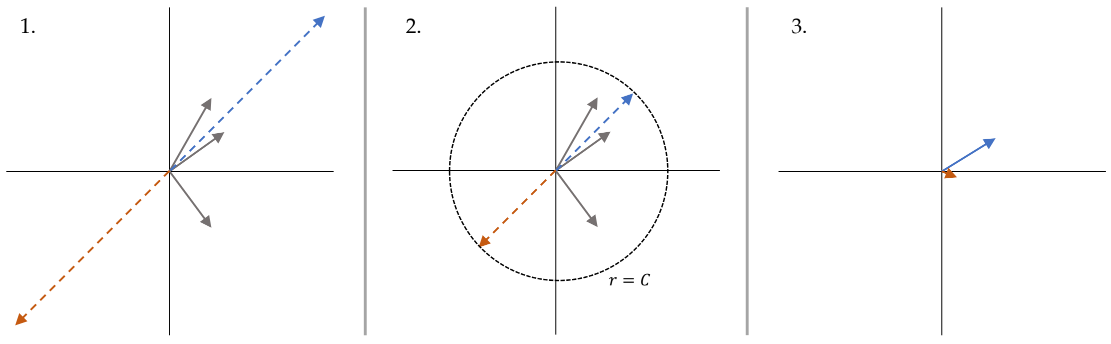


Figure 2.2: Visualization of gradient clipping and averaging. A batch of size $L = 4$ consists of the three gray gradients plus the challenge element x. x is either the blue or the red gradient.
1. Calculate the gradients, 2. Clip gradients to maximal length C, 3. Calculate mean gradient.

Definition 2.6 (l_2-Sensitivity). The l_2-sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \rightarrow \mathbb{R}^k$ is defined as:

$$\Delta_2(f) = \max_{x,y \in \mathbb{N}^{|\mathcal{X}|}, \|x-y\|_1 = 1} \|f(x) - f(y)\|_2$$

With this now bound sensitivity, the average (clipped) gradient is calculated and sufficiently scaled noise can be added to achieve DP in each update step of DP-SGD. To add noise, DP-SGD uses the *Gaussian mechanism*.

Definition 2.7 (Gaussian Mechanism). With $\mathcal{N}(0, S_f^2 \cdot \sigma^2)$ being the normal/Gaussian distribution with mean $\mu = 0$ and standard deviation $S_f \cdot \sigma$, the Gaussian mechanism is defined by

$$\mathcal{M}(x) \triangleq f(x) + \mathcal{N}(0, S_f^2 \cdot \sigma^2)$$

where S_f represents the sensitivity of function f.

The Gaussian mechanism is proven to achieve DP with $\sigma = \sqrt{2 \ln \frac{1.25}{\delta}} / \varepsilon$.

Lemma 2.8 (Privacy of the Gaussian Mechanism). *For an arbitrary $\varepsilon \in (0,1)$ and $c^2 > 2\ln(1.25/\delta)$, the Gaussian Mechanism with parameter $\sigma \geq c\Delta_2(f)/\varepsilon$ is (ε, δ)-differentially private.*

The corresponding proof is found in [DR14]. Algorithm 1 shows the outline of DP-SGD. Figure 2.2 shows a visualization of the gradient clipping and averaging process.
Two factors suggest that adding the described amount of noise to each training step does indeed achieve DP but is the result of highly untight privacy estimations and thus is a

too pessimistic assessment of the actual privacy leakage. First, by design, the challenge element x_c is only sampled in one of the N/L batches of each epoch, therefore only with a sampling probability of $q = L/N$ has to be considered in the scaling of the noise for a given batch. Second, the current estimation ignores how these bounds behave under composition, meaning the repeated execution of a mechanism. The authors of DP-SGD show that under composition the bounds indicated by both sequential composition ([DR14], Theorem 3.16) as well as the advanced/strong composition theorem ([DRV10, DR14]) are still untight and leave room for more accurate estimations of the privacy loss induced by DP-SGD. For this reason, they propose a new privacy accounting technique, the *moments accountant*.

The moments accountant successfully considers both factors, the sampling probability q and the resulting decrease in privacy loss per step as well as how DP-SGD behaves over the composition of several training steps by tracking the moments of the privacy loss random variable. As a result, much tighter bounds for the overall privacy spent can be formulated.

Lemma 2.9 (Bounds of DP-SGD). *There exist constants c_1 and c_2 so that given the sampling probability $q = L/N$ and the number of steps T, for any $\varepsilon < c_1 q^2 T$, Algorithm 1 is (ε, δ)-differentially private for any $\delta > 0$ if we choose*

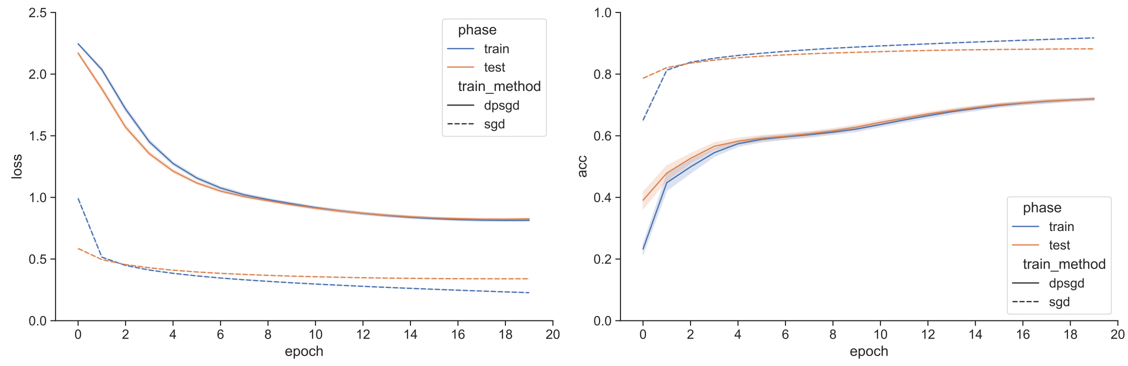
$$\sigma \geq c_2 \frac{q\sqrt{T \ln(1/\delta)}}{\varepsilon}$$

The corresponding proof of this lemma is found in the original paper ([DR16]).

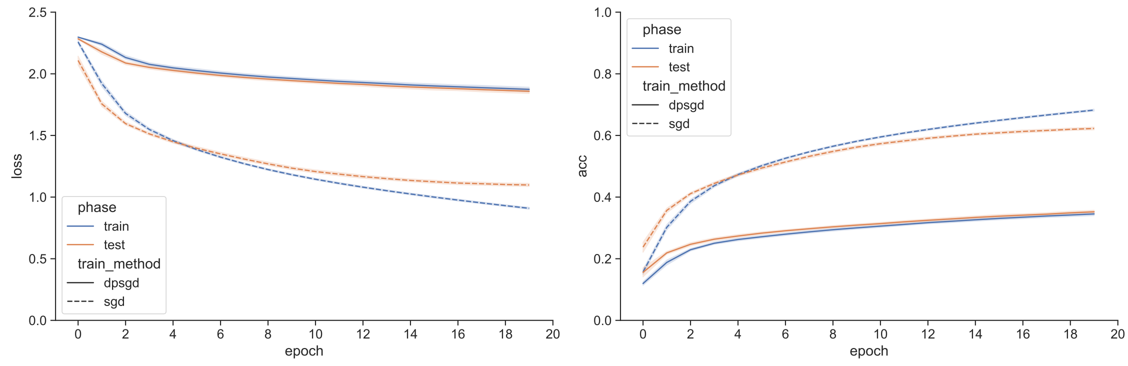
Algorithm 1: DP-SGD (Outline)

```
Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters:
       learning rate $\eta^{(t)}$, noise scale $\sigma$, batch size $L$, gradient norm bound $C$.
Output: The model's parameters after training: $\theta^{(T)}$
1 Initialize $\theta^{(0)}$ randomly
2 for $t \in [T]$ do
3     Take a random sample $L^{(t)}$ with sampling probability $L/N$
4     foreach $i \in L^{(t)}$ do
5         $\boldsymbol{g}^{(t)}(x_i) \leftarrow \nabla_{\theta^{(t)}} \mathcal{L}(\theta^{(t)}, x_i)$                 // Compute gradient
6         $\bar{\boldsymbol{g}}^{(t)}(x_i) \leftarrow \boldsymbol{g}^{(t)}(x_i) / \max(1, \frac{\|\boldsymbol{g}^{(t)}(x_i)\|_2}{C})$   // Clip gradient
7     $\tilde{\boldsymbol{g}}^{(t)} \leftarrow \frac{1}{L}\left(\sum_i \bar{\boldsymbol{g}}^{(t)}(x_i) + \mathcal{N}(0, \sigma^2 C^2 \boldsymbol{I})\right)$        // Add noise
8     $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta^{(t)} \tilde{\boldsymbol{g}}^{(t)}$                              // Descent
```

When using DP-SGD for basic image classification tasks it becomes apparent that even



(a) Training progression of DP-SGD ($\varepsilon = 3.0$) and SGD on the Fashion-MNIST dataset. The model used is described in Listing A.1.



(b) Training progression of DP-SGD ($\varepsilon = 3.0$) and SGD on the Cifar-10 dataset. The model used is described in Listing A.3.

Figure 2.3: Training results of DP-SGD compared to SGD for different datasets. Solid lines represent mean values whereas the semi-transparent area around them represents the 95% confidence interval.

with the tighter bounds given by the moments accountant, DP-SGD suffers from a extensive loss in utility when comparing its results to those of its not-privacy-preserving counterpart SGD. Figure 2.3 shows the training progressions of DP-SGD and SGD for the standard benchmark datasets *Fashion-MNIST* ([XRV17]) and *Cifar-10* ([Kri09]).

Already for $\varepsilon = 3.0$ the achieved accuracy decreases by $\approx 23\%$ for F-MNIST and ca. 50% for the more challenging Cifar-10. Choosing a more practical $\varepsilon \approx 1.0$ the results deteriorate even further. This major drawback motivated the work on this thesis to investigate a possible improvement of DP-SGD's utility while preserving its privacy guarantees.

Remark: For the results shown, a relatively small batch size of $L = 32$ was chosen. The original paper suggests a batch size of $L \approx \sqrt{N}$ with N being the size of the dataset. However, our experiments showed that the utility did not differ when using either batch size for the investigated datasets. Since later sections benefit from smaller batchsizes, we decided to discuss the results for $L = 32$ already in this section.

3 Voted-SGD

The initial goal of this thesis was to investigate whether an additional voting step can be used to reduce the negative impact of additive noise on the performance of a differentially private algorithm. As a case study of such a algorithm, DP-SGD was chosen. This chapter describes the implementation of an additional voting step to DP-SGD, discusses its results and shows how these results motivate a deeper analysis of additive noise.

3.1 Motivation

The idea to add a voting step to existing mechanisms arose when analysing the impact of voting on simple, one-dimensional applications. An example for these simple applications is to determine the median of a given distribution of sampled, one-dimensional data points $\{x_1, \ldots, x_N\}$. To achieve this in a privacy-preserving manner, one could think of the following mechanism: Given N one-dimensional data points called samples $X = \{x_1, \ldots, x_N\}$, the mechanism $\mathcal{M}_{med}$ first calculates the clipped mean of X. Afterward, Gaussian noise is added to this clipped mean to perturb its true value. This perturbed mean is then output as the median of X. We call this approach *noisy-median*. Of course, this approximation of the median via the mean only works sufficiently, if the data is densely packed around a single center or almost evenly distributed. If the data e.g. consists of two far apart clusters, the mean and the median will differ greatly. Nonetheless, this basic example suffices to show the positive effects an additional voting step can have. Additionally, one could argue that many mechanisms in DP often meet these requirements but this lays out of the scope of this thesis and thus this example simply remains as a motivation. A scenario meeting the described requirements are samples taken from a simple Gaussian distribution $\mathcal{N}(\mu = 1., \sigma = 0.2)$.

To add an additional voting step to the noisy-median algorithm, the voters, the candidates as well as the voting criterion need to be defined. In this thesis, whenever applying voting, the unperturbed data is used as voters since the unperturbed execution of the underlying algorithm is assumed to be optimal (at least this is what differentially-private mechanisms must compare to). Candidates are acquired by sampling from the Gaussian distribution not just once but multiple times. As the voting criterion, the *Euclidean distance* is used. Summarizing this scenario, the (unperturbed) input samples vote on a number of samples from the Gaussian distribution representing the perturbed mean. Voters will give their

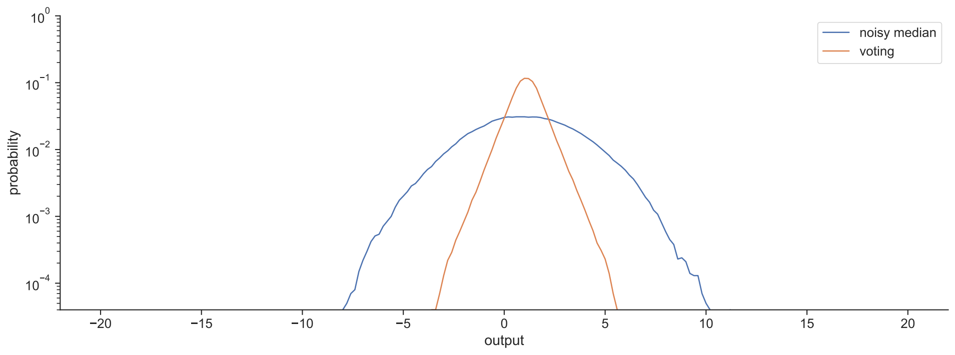


Figure 3.1: Results of median calculation of $\mathcal{N}(1.0, 0.2)$ using *noisy median* compared to the voting approach.

vote to the one sample that has the smallest Euclidean distance to them.

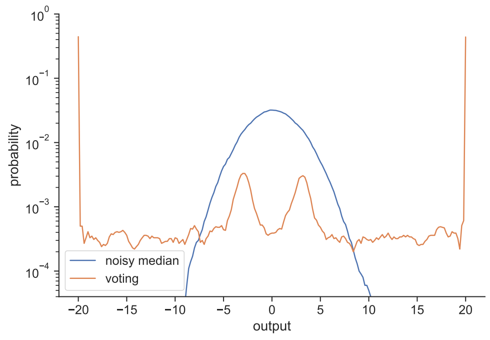
Figure 3.1 shows the resulting output distributions of both the noisy-median as well as the voting approach for calculating the median of $\mathcal{N}(\mu = 1., \sigma = 0.2)$. It is easy to see that both output distributions have a mean of ≈ 1.0 which is the desired result. Though, when comparing their variance, the voting approach has an advantage ($var(\mathcal{M}_{med}) \approx 6.433$ whereas $var(\mathcal{M}_{vote}) \approx 0.642$) which suggests a better reliability of the mechanism with more stable results.

Another advantage of the voting approach becomes apparent when changing the input distribution from one Gaussian distribution to a mixture of two Gaussians with different means, e.g., $\mathcal{N}(-3, 0.2)$ and $\mathcal{N}(+3, 0.2)$. Figure 3.2 shows the results again for both the noisy-median as well as the voting approach. The benefit of the latter is that it suffices to depict the two hills of the input distribution while the noisy-median fails to do so. Both for the current setting of finding a distribution's median as well as for many other applications it is important to get a depiction of the input distribution and its clusters. A supposedly bad property of the voting approach is that a lot of probability mass can be observed at the tails of the output distribution. This causes the left and right peaks in Figure 3.2a and can be seen in more detail in Figure 3.2b. How much this behaviour affects the results of real applications is investigated in later experiments.

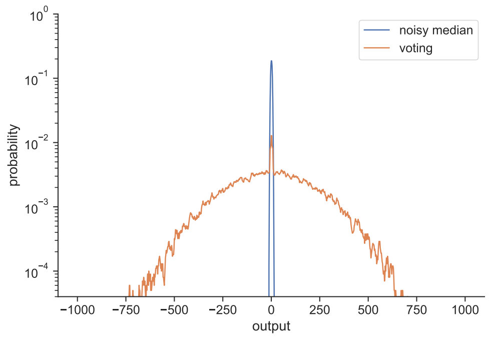
Nonetheless, this first experiment provides promising results which motivates the transfer to more complex mechanisms, e.g., DP-SGD and allows for the following conjecture.

Conjecture 3.1. *An additional voting step can be used to improve the utility of mechanisms using additive noise.*

Whether this conjecture holds or not is investigated in this chapter.



(a) Results for median calculation (zoomed view).



(b) Results for median calculation (complete view).

Figure 3.2: Results of median calculation of $\mathcal{N}(-3.0, 0.2) + \mathcal{N}(3.0, 0.2)$ using *noisy median* compared to the voting approach.

3.2 Implementation

As the name suggests, the main idea of Voted-SGD is to use an additional voting step to improve the results of DP-SGD. It introduces *voters* that vote on several *candidates* to get the one candidate which most likely solves a given objective better than its competitors. This underlying scheme could be applied to any differentially private algorithm that uses additive noise but this thesis focuses on its application to the training of neural networks and analyses the quality of its results compared to DP-SGD.
The general voting approach can be split into three components, independent of the underlying algorithm it is used for.

Voters (general) Unperturbed variables that are assumed to represent the optimal behavior of the underlying algorithm but cannot directly be used without inducing immense privacy leakage. Voters vote on the candidates according to the voting criterion.

Candidates (general) Variables with additional random components that induce limited privacy leakage when being generated or used.

Voting criterion (general) The scale by which the voters decide for what candidates to vote. The voting criterion should enable the voters to find the one candidate that most closely reflects their own influence on the underlying algorithm.

The first step of the implementation of Voted-SGD is to adapt these three components to the training progress of a neural network. As Voted-SGD is an adaptation of DP-SGD, it still focuses on the gradient calculation that is used to update the network's parameters θ in each training step. The resulting components are

Voters (Voted-SGD) The unperturbed gradients of every input sample in a batch.

Candidates (Voted-SGD) The average clipped gradient of a batch with additional Gaussian noise. Each candidate is of form $\boldsymbol{g}_{mean} + \mathcal{N}_i$ with $\boldsymbol{g}_{mean}$ being the average clipped gradient of the batch and $\mathcal{N}_i$ being the ith sample from the added Gaussian noise.

Voting criterion (Voted-SGD) A common metric to measure the similarity between gradients or vectors is the l_2-norm. Another possible metric would be the $\cos$-distance. But since this would ignore any information about the gradients' lengths and by this an important property of the thereto related update length of θ, the l_2-norm is used in Voted-SGD.

With these three components defined, one update step of Voted-SGD has the following structure: As described for SGD as well as DP-SGD the input dataset of size N is first divided into N/L disjoint batches of size L. Then for each batch, the gradients for all input samples in that batch are calculated as well as the clipped mean of these gradients. To compute the n_c candidates of the subsequent voting step, for each candidate, the clipped mean gradient is added to one sample from appropriately scaled Gaussian noise. In the last step, the L unperturbed, unclipped gradients vote on the n_c candidates and the one candidate which gathered the most votes is used to update the network's parameters θ. This process is repeated for all $T = N/L \cdot \#epochs$ update steps.
Algorithm 2 shows the described outline of Voted-SGD as pseudocode.
Algorithm 3 shows the actual voting in more detail where the closest candidate to each voter is computed and the votes per candidate are collected. To preserve DP during the voting process, *Report Noisy Max* (RNM) is used, of which Algorithm 4 shows the outline. RNM is further discussed in the next section dedicated to the privacy guarantees of Voted-SGD.

3.3 Privacy of Voted-SGD

When determining the overall privacy leakage of Voted-SGD two parts have to be considered: the calculation of the n_c candidates as well as the voting process with the publishing of the elected candidate. These two parts can be considered separately if the overall privacy budget ε is split evenly between them. This can be formulated as the following lemma:

Algorithm 2: Voted-SGD (Outline)

```
Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters:
       learning rate $\eta^{(t)}$, noise scale $\sigma'$, group size $L$, gradient norm bound $C$,
       number of candidates $n_c$.
Output: The model's parameters after training: $\theta^{(T)}$.
1  Initialize $\theta^0$ randomly
2  for $t \in [T]$ do
3      Take a random sample $L^{(t)}$ with sampling probability $L/N$
4      foreach $i \in L^{(t)}$ do
5          $\boldsymbol{g}^{(t)}(x_i) \leftarrow \nabla_{\theta^{(t)}}\mathcal{L}(\theta^{(t)}, x_i)$          // Compute gradient
6          $\bar{\boldsymbol{g}}^{(t)}(x_i) \leftarrow \boldsymbol{g}^{(t)}(x_i)/\max(1, \frac{\|\boldsymbol{g}^{(t)}(x_i)\|_2}{C})$          // Clip gradient
7      $\bar{\boldsymbol{g}}^{(t)}_{mean} \leftarrow \frac{1}{L}\sum_i \bar{\boldsymbol{g}}^{(t)}(x_i)$          // Compute mean gradient
8      for $k \in [n_c]$ do
9          $\tilde{\boldsymbol{g}}^{(t)}_k \leftarrow \bar{\boldsymbol{g}}^{(t)}_{mean} + \frac{1}{L}\cdot\mathcal{N}(0, \sigma' C\boldsymbol{I})$          // Compute voting candidates
10     $\hat{\boldsymbol{g}}^{(t)} \leftarrow voting(\tilde{\boldsymbol{g}}^{(t)}, \boldsymbol{g}^{(t)})$          // Compute best candidate
11     $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta^{(t)}\hat{\boldsymbol{g}}^{(t)}$          // Descent
```

Algorithm 3: Voting (Outline)

```
Input: Candidates $\{c_1, \ldots, c_n\}$, voters $\{g_1, \ldots, g_k\}$, noise scale $\sigma_v$.
Output: The most-voted candidate $c_m$.
1  Initialize Vote counts $[v_1, \ldots, v_n]$ as $[0, \ldots, 0]$
2  foreach $g \in \{g_1, \ldots, g_n\}$ do
3      $j \leftarrow \{j | \forall i : \|g - c_j\|_2 \leq \|g - c_i\|_2\}$          // Compute closest candidate
4      $v_j \leftarrow v_j + 1$          // Increase candidate's vote count
5  $m \leftarrow ReportNoisyMax([v_1, \ldots, v_n], \sigma_v)$
6  return $c_m$
```

Algorithm 4: ReportNoisyMax (Outline)

```
Input: Histogram $H$, noise scale $\sigma_v$.
Output: Index $i$ of noisy maximum of histogram.
1  foreach $h_i \in H$ do
2      $c_i \leftarrow h_i + \mathcal{L}(0, \sigma_v)$          // Add Laplace noise to each entry
3  return $\text{argmax}_i c_i$
```

Lemma 3.2. *If both the candidate calculation as well as the voting step of Voted-SGD achieve $(\varepsilon/2, \delta/2)$-DP, Voted-SGD overall achieves (ε, δ)-DP.*

Proof. This lemma directly follows from the sequential composition theorem.

Theorem 3.3 (Sequential Composition ([DR14])). *Let $\mathcal{M}_i : \mathcal{D} \to \mathcal{R}_i$ with domain $\mathcal{D}$ and range $\mathcal{R}_i$ be a $(\varepsilon_i, \delta_i)$-differentially private randomized mechanism for $i \in [k]$. Then if $\mathcal{M}_{[k]} : \mathcal{D} \to \prod_{i=1}^{k} \mathcal{R}_i$ is defined to be $\mathcal{M}_{[k]}(x) = (\mathcal{M}_1(x), \ldots, \mathcal{M}_k(x))$ then $\mathcal{M}_{[k]}$ is $(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i)$-differentially private.*

In other words, when sequentially composing k (ε, δ)-differentially private mechanisms, the result achieves $(k\varepsilon, k\delta)$-DP. As a result, with a total privacy budget of ε, both the candidate calculation as well as the voting process have to achieve $(\varepsilon/2, \delta/2)$-DP. □

For simplicity, from now on the privacy budget of the candidate calculation is called $\varepsilon' = \varepsilon/2$ and the privacy budget of the voting step is called $\varepsilon_v = \varepsilon/2$.
When further analysing the privacy loss of the candidate calculation, it becomes apparent that it can easily be derived from the privacy budget of DP-SGD. The privacy proof of DP-SGD assumes that after each training step, the resulting update gradient is published and thus causes the privacy leakage shown in Lemma 2.9. If for Voted-SGD not only one but n_c update gradients are published, this induces the same privacy leakage as running DP-SGD not over T but over $n_c \cdot T$ training steps. The privacy leakage of Voted-SGD can thus be estimated by the results shown in Lemma 2.9 when instead of T using $T' = T \cdot n_c$ which results in a larger privacy leakage or requires a larger σ to achieve the same privacy goal as DP-SGD for the same number of update steps.
For the voting step, *Report Noisy Max* (RNM) is used to achieve DP. RNM returns the index of the maximal count present in a histogram in a privacy-preserving manner. Since the voting process of Voted-SGD produces a histogram of counts (votes per candidate) and uses the most voted candidate for its update step, RNM is very well suited for this application. [DR14] show that the following lemma holds and thereby that the voting process of Voted-SGD achieves DP.

Lemma 3.4. *Report Noisy Max is $(\varepsilon, 0)$-differentially private when independently generated Laplace noise $Lap(1/\varepsilon)$ is added to each count in a histogram and afterward only the index of the largest noisy count is released.*

Since RNM achieves Pure-DP, Lemma 3.2 can be reformulated as

Lemma 3.5. *If the candidate calculation of Voted-SGD achieves $(\varepsilon' = \varepsilon/2, \delta)$-DP and the voting step achieves $(\varepsilon_v = \varepsilon/2, 0)$-DP, Voted-SGD overall achieves (ε, δ)-DP.*

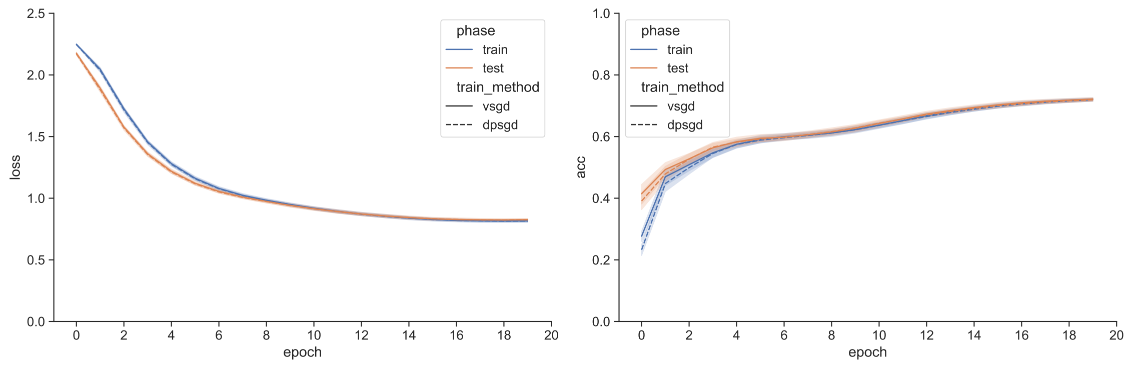
The correctness again directly follows from the sequential composition theorem.

It has to be noted that the privacy bounds shown are still very loose and require further research to more accurately capture the true privacy leakage of Voted-SGD. Especially the results shown by [BBG20] proof tighter bounds for subsampling algorithms like Voted-SGD via *privacy profiles* and could be extended to capture the influence of compositions. Since this more sophisticated approach lies outside the scope of this thesis, this is left for future research. Nonetheless, for this thesis, the results of this section suffice to show that Voted-SGD achieves DP and enables to produce comparable results for DP-SGD and Voted-SGD.

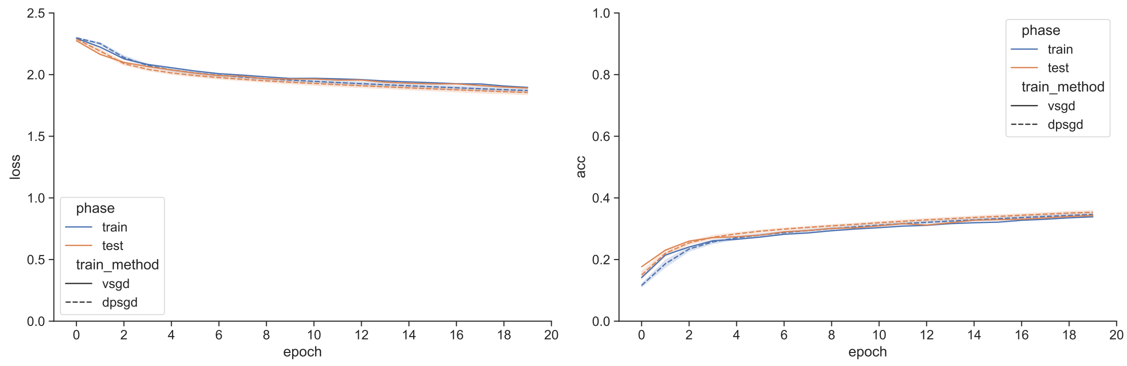
3.4 Results

To test whether Voted-SGD is a suitable alternative to DP-SGD, it is used to perform basic image classification tasks both on the *Fashion-MNIST* as well as the *Cifar-10* dataset. For this first test, the number of candidates in each batch of size L is set to $n_c = L/2$. The privacy parameters σ', C and σ_v are derived from the parameters of DP-SGD as described in the last section. Thereby $C = C_{dpsgd}$, $\sigma' = 2 \cdot \sqrt{n_c} \cdot \sigma_{dpsgd}$ and $\sigma_v = 2 \cdot \sigma_{dpsgd}$ (cf. Algorithms 2-4 for the implementation of these parameters; parameters with a lower $dpsgd$ are those used in Algorithm 1). Figure 3.3 shows the respective results of Voted-SGD and DP-SGD for comparison.

It becomes apparent that Voted-SGD does not suffice to outperform the utility of DP-SGD but that both algorithms perform very equally. To understand why this is, the actual voting process is further examined. As a first step, the grade of agreement across the voters is investigated. To benefit from a voting process, across all candidates there has to exist ideally one candidate that represents the best option for all or at least most voters. If all candidates are equally favorable for the voters, the voting process has no benefit over just choosing one of the candidates randomly. A measure for the voter's agreement is to calculate the entropy of the voting result which is a histogram collecting the number of votes per candidate. Figure 3.4 shows how the voting result's entropy develops over the course of the training's epochs. For a batch size of $L = 32$, $n_c = 16$ is chosen resulting in a maximum entropy of $H(X) = -\sum_{i=1}^{n_c} p(x_i)\log(p(x_i)) \approx 2.773$ (where X represents the candidates) if all candidates are equally voted for, and a minimum entropy of $H(X) = 0$ if only one candidate is voted for. Figure 3.4 shows that for the first epochs often a clear best option appears to exist resulting in a high agreement across the voters and thereby a small entropy of the voting result. Unfortunately, this changes rather quickly ending in a much higher entropy at approximately epoch 7. From there on, until the end of the training, the entropy basically remains the same. This entropy development suggests,



(a) Training progression of Voted-SGD and DP-SGD on the Fashion-MNIST dataset. The model used is described in Listing A.1.



(b) Training progression of Voted-SGD and DP-SGD on the Cifar-10 dataset. The model used is described in Listing A.3.

Figure 3.3: Training results of Voted-SGD compared to DP-SGD for different datasets and $\varepsilon = 3.0$. Solid lines represent mean values whereas the semi-transparent area around them represents the 95% confidence interval.

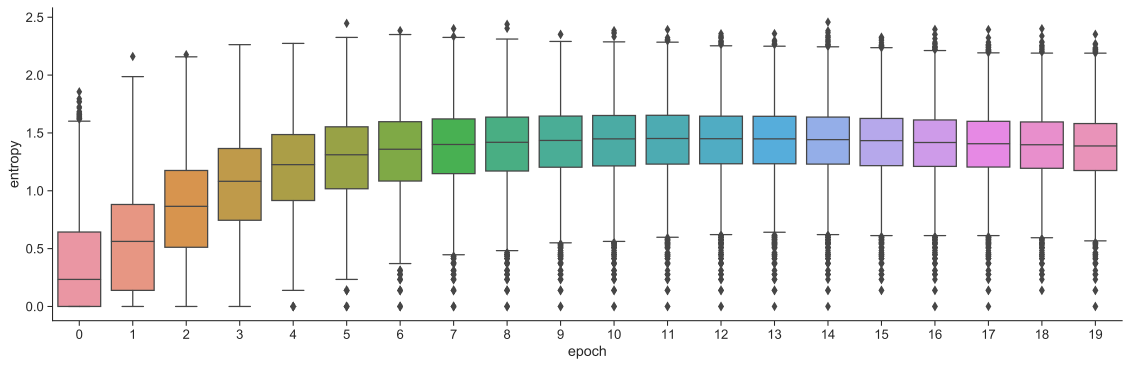
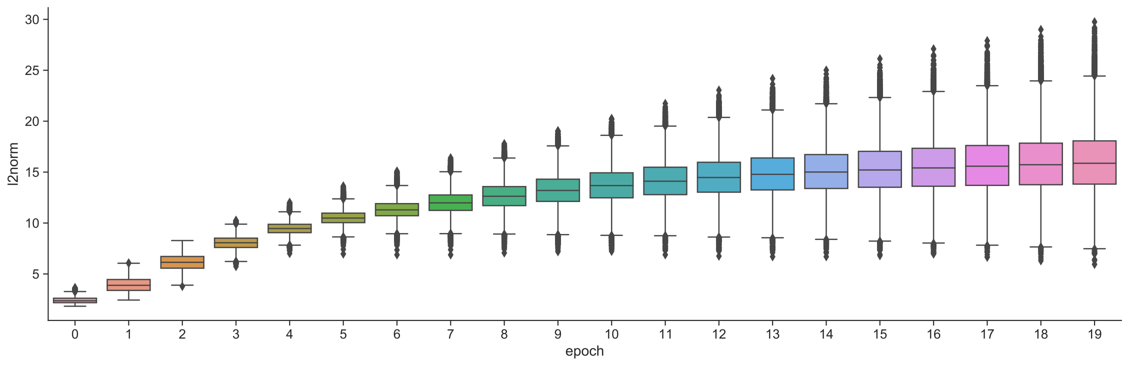


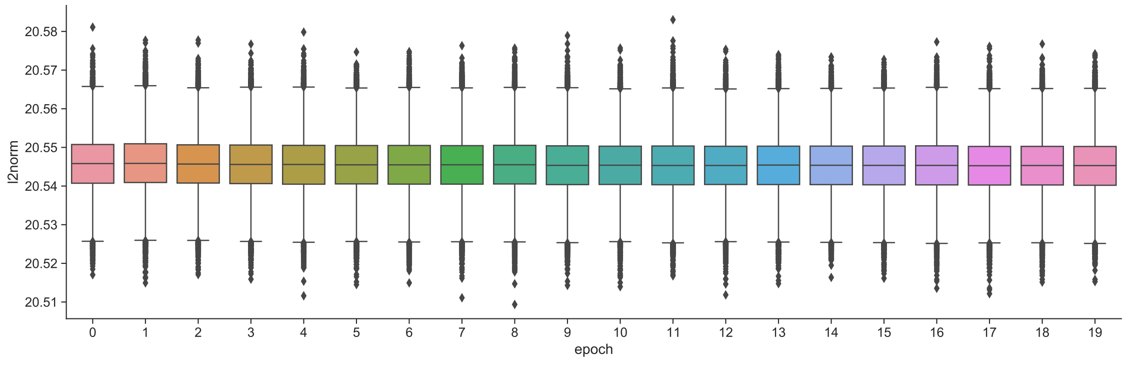
Figure 3.4: Entropy of the voting result over the course of training on Fashion-MNIST.

that for the most part of the training, the voting process does not suffice to produce one clearly favorable candidate but instead suggests that the optimal gradient to update the network's parameters θ is different for many of the voters.
The dissent of voters can be explained by two scenarios: Either the candidates resemble the voters too close, providing most voters with their own personal favorite for which they vote in the voting process; or the candidates and the voters differ too much so that each voter has to decide between only bad options and pick their least evil. To determine which scenario applies to the voting process of Voted-SGD, Figure 3.5 shows a comparison between the l_2-norm of the voters and the candidates.
From this comparison, it becomes apparent that the reason for the missing improvement through the voting process is most likely the second scenario: voters and candidates have very distinct properties leaving the voters with the choice of the least evil. Not only are the lengths (l_2-norm) of the candidates very different from those of the voters, but also appear to be very uniform in this length. With the two farthest outliers being of length 20.51 and 20.58 respectively, their l_2-norm at most varies in the second decimal place while having an average length of approximately 20.545. This essentially leaves the voters no choice about the size of the network's next update step. A second factor to be considered is that the candidates shown in Figure 3.5 result from clipping the length of the unperturbed gradients of all batches at length $C = 1.0$. That the addition of noise to gradients with a maximal length of 1.0 results in gradients of length > 20, shows the immense share random noise has in these candidates. This further strengthens the assumption that the voters have only a limited influence on the network's updates.

The findings presented in this section suggest that random noise has a dominant influence on the results produced by mechanisms like Voted-SGD that aim to achieve DP. This insight motivates the second part of this thesis starting with the next chapter which is



(a) l_2-norm distribution of the voters.



(b) l_2-norm distribution of the candidates.

Figure 3.5: l_2-norm distribution of voters and candidates over the training on the Fashion-MNIST dataset.

dedicated to a deeper analysis of said noise and its influence on differentially private mechanisms.

4 Analysis of Additive Noise in DP

The results of the last chapter call for a deeper analysis of the noise being added by the sensitivity method. At this point, two main questions arise from the previous results: Firstly, is a more accurate description of the noise's properties possible, which explains the l_2-norms seen in Figure 3.5? Secondly, the question arises whether the poor performance with regards to utility can be explained by the properties of additive noise. The answer to these two questions is the main result of this thesis.

4.1 Theoretical Foundations

To get a general description of the noise being added, independent of the underlying application, the following lemma can be formulated.

Lemma 4.1. *The noise being added by the Gaussian mechanism can be described as a d-variate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and positive-definite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ where d is the number of dimensions of the data that is being perturbed.*

The Gaussian mechanism adds independent noise with mean μ and variance σ^2 to each dimension of the data if the covariance $\boldsymbol{\Sigma}$ is a diagonal matrix. Thus, to prove that the described lemma holds, it suffices to prove the following lemma.

Lemma 4.2. *The probability density function (PDF) of d one-dimensional Gaussian distributions $\mathcal{N}(\mu_i, \sigma_i^2)$ with mean μ_i and standard deviation σ_i where $i \in \{1, \dots, d\}$ is equal to the PDF of one d-variate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean vector*

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_d \end{bmatrix}$$

and covariance matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0.0 & \cdots & 0.0 \\ 0.0 & \sigma_2^2 & \cdots & 0.0 \\ \vdots & \vdots & \ddots & \vdots \\ 0.0 & 0.0 & \cdots & \sigma_d^2 \end{bmatrix},$$

In other words,

$$pdf_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\boldsymbol{x}) = \prod_{i=1}^{d} pdf_{\mathcal{N}(\mu_i,\sigma_i^2)}(x_i)$$

Proof. The PDF of a multivariate Gaussian distribution is defined as

$$pdf_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\boldsymbol{x}) = \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2}\right)$$

Since the covariance matrix $\boldsymbol{\Sigma}$ is a diagonal matrix, its inverse $\boldsymbol{\Sigma}^{-1}$ is again a diagonal matrix with the reciprocal values of $\boldsymbol{\Sigma}$ on its diagonal.

$$\Sigma^{-1} = \begin{bmatrix} \frac{1}{\sigma^2} & 0.0 & \cdots & 0.0 \\ 0.0 & \frac{1}{\sigma^2} & \cdots & 0.0 \\ \vdots & \vdots & \ddots & \vdots \\ 0.0 & 0.0 & \cdots & \frac{1}{\sigma^2} \end{bmatrix}$$

With these definitions, the PDF can be described as

$$\begin{aligned} pdf_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\boldsymbol{x}) &= \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2}\right) \\ &= \frac{1}{\sqrt{(2\pi)^d \prod_{i=1}^{d}\sigma_i^2}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2}\right) \\ &= \frac{1}{\sqrt{(2\pi)^d \prod_{i=1}^{d}\sigma_i^2}} \exp\left(-\frac{1}{2}\begin{bmatrix}(x_1-\mu_1)\cdots(x_d-\mu_d)\end{bmatrix}\begin{bmatrix}\frac{1}{\sigma_1^2}(x_1-\mu_1) \\ \vdots \\ \frac{1}{\sigma_d^2}(x_d-\mu_d)\end{bmatrix}\right) \\ &= \frac{1}{\sqrt{(2\pi)^d \prod_{i=1}^{d}\sigma_i^2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{1}{\sigma_i^2}(x_i-\mu_i)^2\right) \\ &= \frac{1}{\sqrt{(2\pi)^d \prod_{i=1}^{d}\sigma_i^2}} \exp\left(-\sum_{i=1}^{d}\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\right) \end{aligned}$$

To complete the proof, it is shown that the same description holds for the PDF of d one-dimensional Gaussian distributions.

$$\begin{aligned}
\prod_{i=1}^{d} pdf_{\mathcal{N}(\mu_i,\sigma_i^2)}(x_i) &= \prod_{i=1}^{d} \frac{1}{\sigma_i\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2\right) \\
&= \frac{1}{\prod_{i=1}^{d}\sigma_i\sqrt{2\pi}} \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2\right) \\
&= \frac{1}{\sqrt{(2\pi)^d\prod_{i=1}^{d}\sigma_i^2}} \exp\left(-\sum_{i=1}^{d}\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\right) \\
&= pdf_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\boldsymbol{x})
\end{aligned}$$

□

With this equality proven, findings for both settings, d independent, one-dimensional Gaussian distributions as well as one d-variate Gaussian distribution, apply to the additive Gaussian noise of the Gaussian mechanism.

4.2 Properties of Additive Gaussian Noise in DP

To achieve the main result of this thesis, the description of additive noise as d independent, one-dimensional Gaussian distributions can be used to determine the estimated mean of the l_2-norm of samples from said additive noise distribution. For this purpose, the following lemma is formulated and proven.

Lemma 4.3. *The mean l_2-norm of Gaussian noise added by the sensitivity method increases with $\sigma \cdot \sqrt{2} \cdot \frac{\Gamma((d+1)/2)}{\Gamma(d/2)}$ where d represents the number of dimensions and σ describes the standard deviation used in each dimension of this Gaussian noise.*

Proof. Let $Z_1, \ldots, Z_d$ be independent, normally distributed random variables with mean 0 and standard deviation 1. Then, the statistic

$$Y = \sqrt{\sum_{i=1}^{d} Z_i^2}$$

is distributed according to the χ-distribution. The mean of the χ-distribution is given by

$$\mu_\chi = \sqrt{2} \cdot \frac{\Gamma((d+1)/2)}{\Gamma(d/2)}$$

where Γ describes the Gamma function.
As described for the Gaussian mechanism, $Z_1', \ldots, Z_d'$ are independent, normally distributed random variables with mean 0 and standard deviation σ. To adjust for the stan-

dard deviation σ, $Z'_1, \dots, Z'_d$ have to be normalized by σ. Thus, with $Z_i = Z'_i\sigma$, a statistic Y' can be formulated as

$$\begin{aligned} Y' &= \sqrt{\sum_{i=1}^{d} Z'^2_i} \\ &= \sqrt{\sum_{i=1}^{d} (Z_i\sigma)^2} \\ &= \sqrt{\sigma^2 \sum_{i=1}^{d} Z_i^2} \\ &= \sigma \cdot \sqrt{\sum_{i=1}^{d} Z_i^2} \\ &= \sigma \cdot Y \end{aligned}$$

Which results in a new mean value and thereby completes the proof of Lemma 4.3.

$$\begin{aligned} \mu'_\chi &= \sigma \cdot \mu_\chi \\ &= \sigma \cdot \sqrt{2} \cdot \frac{\mathit{\Gamma}((d+1)/2)}{\mathit{\Gamma}(d/2)} \end{aligned} \qquad \square$$

This first result shows that the l_2-norm of samples from a d-variate Gaussian distribution is determined by two factors: the dimensionality of the distribution and thus by the output dimensionality of the computation that the Gaussian mechanism is used for to perturb, as well as the standard deviation used in each of its dimensions. Since the l_2-norm is direction-independent (in relation to μ), these results additionally suggest that the PDF of the described d-variate Gaussian noise is the surface of a d-sphere or, figuratively speaking, a d-dimensional *ring* with a mean radius of $r = \mu'_\chi$ and center μ.

To more accurately describe the structure of this noise ring, another description is given by the PDF of a d-variate Gaussian distribution. For this purpose, a random variable D is defined, which describes the l_2-norm of samples from a d-variate Gaussian distribution. The following equation describes the probability of the l_2-norm of such samples to be of value u.

$$\begin{aligned}\Pr[D=u] &= \int_{\forall \boldsymbol{w}:\|\boldsymbol{w}\|=u} pdf_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\boldsymbol{w})d\boldsymbol{w}\\ &= \int_{\forall \boldsymbol{w}:\|w\|=u} \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}}\exp\left(-\frac{(\boldsymbol{w}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{w}-\boldsymbol{\mu})}{2}\right)d\boldsymbol{w}\end{aligned}$$

As before, only zero-centered distributions are considered, so $\boldsymbol{\mu}=[0,\ldots,0]$.

$$\begin{aligned} &= \int_{\forall \boldsymbol{w}:\|\boldsymbol{w}\|=u} \frac{1}{\sqrt{(2\pi)^d}\sigma^d}\exp\left(-\frac{\boldsymbol{w}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{w}}{2}\right)d\boldsymbol{w}\\ &= \int_{\forall \boldsymbol{w}:\|\boldsymbol{w}\|=u} \frac{1}{\sqrt{(2\pi)^d}\sigma^d}\exp\left(-\frac{\|\boldsymbol{w}\|^2}{2\sigma^2}\right)d\boldsymbol{w}\\ &= \int_{\forall \boldsymbol{w}:\|\boldsymbol{w}\|=u} \frac{1}{\sqrt{(2\pi)^d}\sigma^d}\exp\left(-\frac{u^2}{2\sigma^2}\right)d\boldsymbol{w}\end{aligned}$$

$\int_{\forall \boldsymbol{w}:\|\boldsymbol{w}\|=u} dw$ describes the surface of a d-dimensional sphere, a d-sphere with radius u. This surface is denoted by $S_{d-1}(R)$ where R is the radius of the d-sphere. This surface is universally defined as $S_{d-1}(R)=\frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}R^{d-1}$.

$$\begin{aligned} &= S_{d-1}(u)\cdot\frac{1}{\sqrt{(2\pi)^d}\sigma^d}\exp\left(-\frac{u^2}{2\sigma^2}\right)\\ &= \frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}\cdot u^{d-1}\cdot\frac{1}{\sqrt{(2\pi)^d}\sigma^d}\exp\left(-\frac{u^2}{2\sigma^2}\right)\\ &= \frac{2}{\Gamma(\frac{d}{2})}\cdot u^{d-1}\cdot\frac{1}{\sqrt{2^d}\sigma^d}\exp\left(-\frac{u^2}{2\sigma^2}\right)\\ &= \frac{1}{2^{(d/2)-1}\cdot\Gamma(\frac{d}{2})}\cdot\frac{u^{d-1}}{\sigma^d}\cdot\exp\left(-\frac{u^2}{2\sigma^2}\right)\end{aligned}$$

This last line corresponds to the PDF of the χ-distribution which is the expected result when considering independent, zero-centered Gaussian distributions as it is done here. This implies several important insights. Firstly, this again underpins that Lemma 4.2 is indeed correct, otherwise, the usage of the PDF of an d-variate Gaussian distribution should yield other results than the χ-distribution. Secondly, the outcome affirms the assumption that the shape of additive noise is a d-dimensional ring or more precisely the surface of a n-sphere as it arises in the shown equations. Lastly, it can be derived that this ring shape applies not only to Gaussian noise but to any noise distribution: The resulting $S_{d-1}(R)$ is independent of the used PDF and thus applies to any distribution. With these results, we conjecture the following statement

Conjecture 4.4. *For any noise distribution, the main share of the (probability) mass of this noise will lie on the surface of a d-dimensional hypersphere. Its radius is determined by the number of dimensions d and parameters specific to the noise distribution. We call this surface the* noise ring.

For the Gaussian distribution we can further define this noise ring.

Lemma 4.5. *The noise ring produced by a zero-centered d-variate Gaussian distribution as used by the Gaussian mechanism with standard deviation σ in each dimension is described by the PDF of the χ-distribution*

$$pdf_{\chi}(u) = \frac{1}{2^{(d/2)-1} \cdot \Gamma(\frac{d}{2})} \cdot \frac{u^{d-1}}{\sigma^d} \cdot \exp\left(-\frac{u^2}{2\sigma^2}\right)$$

4.3 Evaluation of Theoretical Results

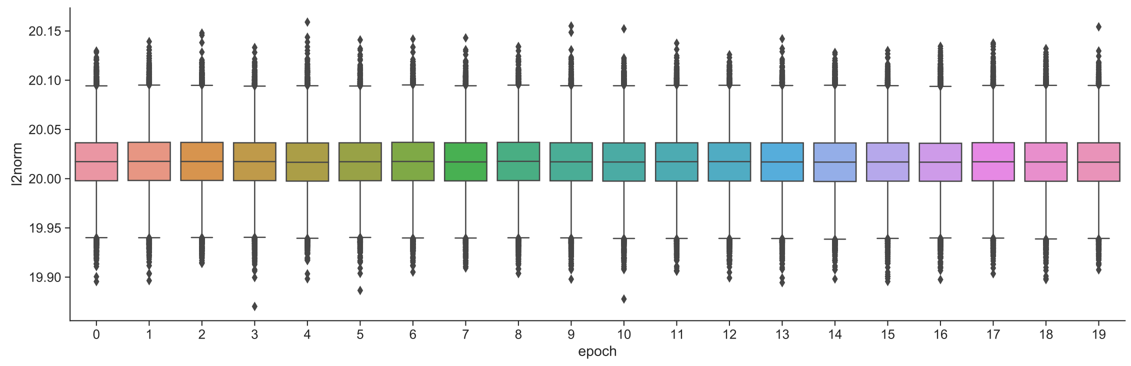
After successfully deriving theoretical descriptions of the length distribution (l_2-norm) of multivariate Gaussian noise, these are compared to the previous practical results to confirm their implications.

As a first step, the results of Lemma 4.3 are compared to those results observed for the implementation of DP-SGD. In these implementations the dimensionality d is defined by the dimensionality of the network used for training or more accurately, by the number of trainable parameters in this network. All networks together with their number of parameters used in the work on this thesis are listed in Appendix A.1.

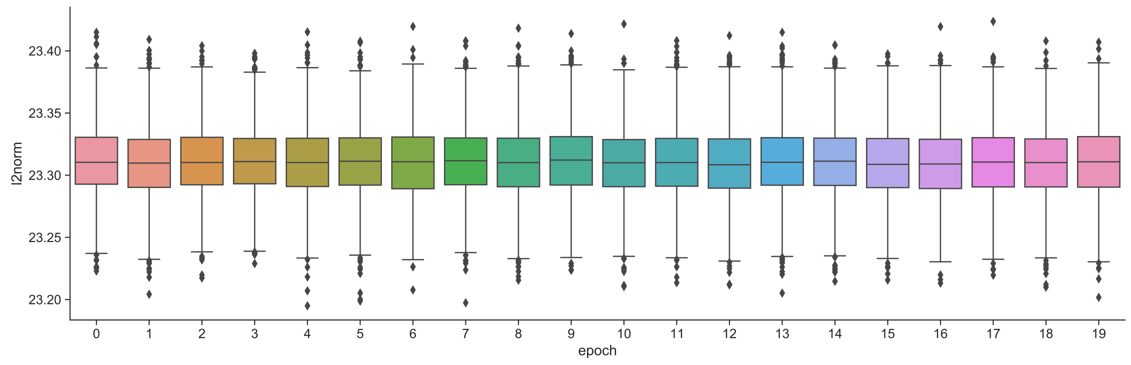
For simplicity, DP-SGD is used to train the respective model on the *Fashion-MNIST* dataset with fixed parameters (cf. Algorithm 1): $C = 1.0, \sigma_{dpsgd} = 1.3, L = 32$. With these parameters, the expected l_2-norm of the additive Gaussian noise results to:

Choosing the model to be *MNIST_FC4_Net* (cf. Listing A.1):

$$\begin{aligned} \sigma \cdot \sqrt{2} \cdot \frac{\Gamma((d+1)/2)}{\Gamma(d/2)} &= \frac{C \cdot \sigma_{dpsgd}}{L} \cdot \sqrt{2} \cdot \frac{\Gamma((d+1)/2)}{\Gamma(d/2)} \\ &= \frac{1.0 \cdot 1.3}{32} \cdot \sqrt{2} \cdot \frac{\Gamma((242762+1)/2)}{\Gamma(242762/2)} \\ &\approx 20.0163 \end{aligned}$$



(a) l_2-norm distribution of resulting gradients for *MNIST_FC4_Net* (cf. Listing A.1).



(b) l_2-norm distribution of resulting gradients for *MNIST_FC7_Net* (cf. Listing A.2).

Figure 4.1: l_2-norm distribution of resulting gradients for DP-SGD over the training on the Fashion-MNIST dataset.

Choosing the model to be *MNIST_FC7_Net* (cf. Listing A.2):

$$\begin{aligned}\sigma \cdot \sqrt{2} \cdot \frac{\Gamma((d+1)/2)}{\Gamma(d/2)} &= \frac{C \cdot \sigma_{dpsgd}}{L} \cdot \sqrt{2} \cdot \frac{\Gamma((d+1)/2)}{\Gamma(d/2)} \\ &= \frac{1.0 \cdot 1.3}{32} \cdot \sqrt{2} \cdot \frac{\Gamma((329226+1)/2)}{\Gamma(329226/2)} \\ &\approx 23.3099\end{aligned}$$

Figure 4.1 shows the resulting l_2-norm distribution of the resulting gradients for the described models and parameters.

These results show two important insights: First, the results of Lemma 4.3 are indeed correct and reproducible through practical implementations. Secondly, they affirm the immense impact that additive noise has on differentially private applications. Figure 4.1 not only shows the l_2-norm of the noise samples, but the resulting gradients comprised of noise added to the clipped mean of unperturbed gradients used by DP-SGD to update the networks parameters. That the norms of these noisy gradients match the expected norms of only noise samples with negligible variation, demonstrates that the influence that unperturbed data has on the gradients is completely dominated by the noise added. By that, the poor performance of DP-SGD as well as Voted-SGD can be explained.

4.4 Generalizability of Findings

To this point, only Gaussian noise was examined due to the implementation of the Gaussian mechanism in DP-SGD and Voted-SGD. Whether the results shown also apply to other noise distribution is analyzed in this section.

As an example, a technique proposed by [CMS11] is considered which can also be used to achieve gradient updates in SGD in a privacy-preserving manner. Unlike for the Gaussian mechanism, in this approach, noise is not added independently in each dimension of the to be perturbed gradients but instead consists of two parts: First, the direction of a noise vector is chosen randomly, e.g., by sampling a vector of length 1 from a d-dimensional Gaussian distribution. Afterward, the length of this noise vector is scaled according to the sensitivity of the mechanism to achieve DP. As the length determining distribution, a 1-dimensional Gamma distribution is used. The resulting noise vector with a random direction and a length sampled from a Gamma distribution e.g., is then added to the clipped mean gradient of a batch in (DP-)SGD.

It has to be noted that this is only a rough sketch of the approach by [CMS11] as a more detailed description lies outside the scope of this thesis. The interested reader is referred to the original paper.

The main questions to be answered is whether this different approach of additive noise does also result in a noise ring establishing for multiple dimensions and whether the radius of this ring increases with dimensionality d. For this purpose, the described mechanism is implemented and the l_2-norm distribution of its resulting noise vectors is analyzed for different d.

Figure 4.2 shows the resulting l_2-norm distributions for different d in comparison to the results produced by the Gaussian mechanism parameterized for the same privacy guarantee ε. The results show that also for this different approach of additive noise, a noise ring establishes with a radius increasing with dimensionality d. This radius increases linearly with d and thus even faster compared to the results of the Gaussian distribution which increases by a factor of $\frac{\Gamma((d+1)/2)}{\Gamma(d/2)}$.

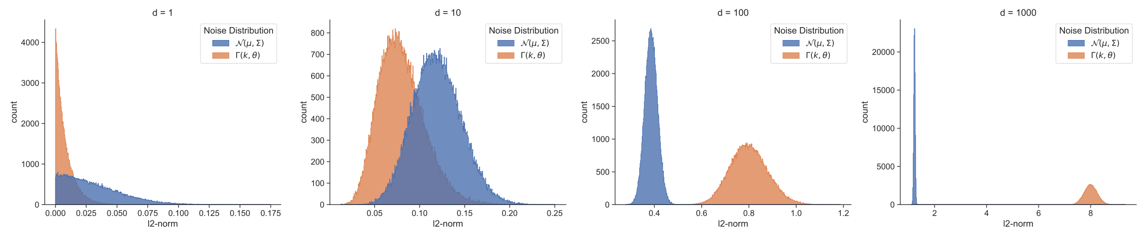


Figure 4.2: l_2-norm distribution of the d-variate Gaussian noise compared to the approach of [CMS11] using a Gamma distribution.

4.5 Results and Implications

With the presented insights of this chapter, the question arises about what these results imply. Can parts of the poor performance (with regards to its utility) of DP-SGD and other differentially-private mechanism using additive noise be explained by the ring-form of the added noise? Does this also explain the missing utility improvement of Voted-SGD? And lastly, can this ring be avoided?

To address the first question, it suffices to discuss what the desired outcome of additive noise is to preserve as much utility as possible. Ideally, the added noise has no influence on the underlying algorithm. For zero-centered noise and the exemplary application of gradient updates in DP-SGD, this means that noise samples close to the noise distributions mean 0 are desirable. Although this is with high-probability achieved in the 1-dimensional case, the probability of this preferred outcome decreases with each additional dimension because of the establishing noise ring. As shown by the results of the previous sections, for ML applications with several thousand of dimensions, no probability mass is left in the noise distribution's mean, making the desirable result impossible.

The same argumentation applies to the explanation of Voted-SGD's poor results. As described in Chapter 3, the standard variation σ has to increase by a factor of $2 \cdot \sqrt{n_c}$ (n_c being the number of candidates) compared to DP-SGD to compensate for the additional privacy leakage. This further increases the radius of the noise ring from which the candidates are sampled. Even if the voting successfully determines the best of those candidates (with regard to utility), this candidate is still considered to have worse properties than all samples that would result from DP-SGD initially. Thus, Voted-SGD is not able to improve the results of DP-SGD.

To further investigate on the implications of this thesis' results for ML applications it can be worthwhile to take a second look at the parameters which determine the noise ring's radius. As already shown, this radius is given by a d-dependent term multiplied by the per dimension standard deviation σ. For DP-SGD this is given by Lemma 2.9. With sampling probability $q = L/N$ and the total number of steps $T = N/L \cdot \#epochs$ it yields the following result.

$$\begin{aligned}
\sigma &\geq c_2 q \frac{\sqrt{T \ln(1/\delta)}}{\varepsilon} \\
&= c_2 \frac{L}{N} \sqrt{\frac{N}{L}} \frac{\sqrt{\ln(1/\delta)}}{\varepsilon} \cdot \sqrt{\#epochs} \\
&= c_2 \frac{L}{\sqrt{L}} \frac{\sqrt{N}}{N} \frac{\sqrt{\ln(1/\delta)}}{\varepsilon} \cdot \sqrt{\#epochs} \\
&= c_2 \sqrt{L} \frac{1}{\sqrt{N}} \frac{\sqrt{\ln(1/\delta)}}{\varepsilon} \cdot \sqrt{\#epochs} \\
&= c_2 \sqrt{\frac{L}{N}} \frac{\sqrt{\ln(1/\delta)}}{\varepsilon} \cdot \sqrt{\#epochs}
\end{aligned}$$

These results suggest that a smaller batch size L or training for fewer epochs could suffice to limit the effect of the noise ring. Unfortunately, these parameters cannot be changed arbitrarily. Often, a certain number of epochs is necessary to achieve sufficient training results. Additionally, lowering the batch size L would increase the per-gradient noise as seen in Algorithm 1 and thus should not be too small. *VGG-16* as proposed by [SZ15] serves as an example for current models used in ML. *VGG-16* consists of $\approx 130mil.$ parameters which results in a d-dependent factor of $\sqrt{2} \cdot \frac{\Gamma((d+1)/2)}{\Gamma(d/2)} \approx 11,400$. Such a big factor, realistically, could never be compensated by the additional factor σ. This suggests that DP-mechanisms like DP-SGD fail to train current ML models.
Lastly, the question remains of whether the emergence of the described noise ring could be prevented. At least with the current approaches using additive noise to achieve DP this is not possible as it is an inherent problem of such noise. The privacy guarantees of DP only hold if the mechanism's results of neighboring inputs only differ at most by $\exp(\pm\varepsilon)$. This implies changes in every of the output's dimensions and thereby requires noise to be added to each of these, be it directly (e.g., as for the Gaussian mechanism) or indirectly (e.g., the approach of [CMS11]). With noise in every dimension, the described noise ring emerges thus making it an inevitable problem of additive noise that strongly affects the performance of differentially private mechanisms and should be considered in any future research on such approaches.

5 Conclusion

The original goal of this thesis was to investigate whether an additional voting step could be used to improve the utility of differentially private mechanisms that use additive noise. Although an improvement of DP-SGD, which was chosen as a case study of such a mechanism, did not result in the desired improvement, an even more valuable finding was discovered: We found an inherent problem of multidimensional noise, namely that at high dimensionality, noise distributions take the form of a multidimensional ring whose radius increases with the number of dimensions.
This insight not only explains why many differentially private mechanisms suffer a great loss in utility compared to not-privacy-preserving algorithms, but also indicates limits to the practicality of these algorithms for high-dimensional applications.
We were able to accurately describe the shape of this ring for the d-variate Gaussian distribution as used in the Gaussian mechanism, and show results that hold for arbitrary noise distributions.
Since, to our knowledge, these results have not been shown by other work, the outcome of this thesis is relevant to many developments in DP and should steer future research in this area.

6 Future Work

The results of this thesis have many implications relevant to future research in DP. This chapter should serve as an overview for possible next steps.

6.1 The Voting Approach

Although the results of this thesis show that an additional voting step is not sufficient to improve the utility of high-dimensional applications, its utility should still be analyzed for low-dimensional settings. For example, (DP-)clustering has many low-dimensional applications and could be improved with this extension.

6.2 Noise in High Dimensions

6.2.1 Further Analysis

With regards to the description of additive noise and the resulting noise ring, this thesis focuses mainly on the Gaussian distribution motivated by the Gaussian mechanism. Although many implications already apply to any other distribution, future research should focus on a more general description of the noise ring described, independent of the noise distribution used.
During the work on this thesis, it became apparent that the radius of the noise ring can be approximated by $\sigma \cdot \sqrt{d}$, at least for the Gaussian distribution. Thus, the d-dependent term $\sqrt{2} \cdot \frac{\Gamma((d+1)/2)}{\Gamma(d/2)}$ derived in this thesis could be approximated by $\sqrt{d}$. Future work should prove that this approximation holds and whether it is valid in general or only for the Gaussian distribution.
From the description of the noise ring and, in particular, its radius, it might be possible to derive upper bounds on the utility of a DP-mechanism as a function of dimensionality. Research in this direction could provide valuable results.

6.2.2 Alternatives

This thesis has shown that the resulting noise ring is an inherent problem of additive noise. Future research should focus on possible alternatives to approaches using additive noise or work on techniques that reduce the dimensionality of DP applications. For ML applications, for example, recent work by [KM21] has provided promising results on avoiding large networks for DP mechanisms. A first, less challenging approach might be to use *convolutional neural networks* instead of DNNs, as used in this thesis, since they generally involve fewer parameters and thus a lower dimensionality for the privacy mechanism.

A Appendix

A.1 Models

Listing A.1: Example of 4-layer fully connected neural network: *MNIST_FC4_Net*

```
Batch size: 32
Input shape: (32, 28, 28)

==============================================================================
Layer (type:depth-idx)                    Output Shape              Param #
==============================================================================
MNIST_FC4_Net                             --                        --
--Linear: 1-1                             [32, 256]                 200,960
--Linear: 1-2                             [32, 128]                 32,896
--Linear: 1-3                             [32, 64]                  8,256
--Linear: 1-4                             [32, 10]                  650
==============================================================================
Total params: 242,762
Trainable params: 242,762
Non-trainable params: 0
Total mult-adds (M): 7.77
==============================================================================
Input size (MB): 0.10
Forward/backward pass size (MB): 0.12
Params size (MB): 0.97
Estimated Total Size (MB): 1.19
==============================================================================
```

Listing A.2: Example of 7-layer fully connected neural network: *MNIST_FC7_Net*

```
Batch size: 32
Input shape: (32, 28, 28)

==============================================================================
Layer (type:depth-idx)                    Output Shape              Param #
==============================================================================
MNIST_FC7_Net                             --                        --
--Linear: 1-1                             [32, 256]                 200,960
--Linear: 1-2                             [32, 256]                 65,792
--Linear: 1-3                             [32, 128]                 32,896
```

```
--Linear: 1-4                           [32, 128]                 16,512
--Linear: 1-5                           [32, 64]                  8,256
--Linear: 1-6                           [32, 64]                  4,160
--Linear: 1-7                           [32, 10]                  650
==============================================================================
Total params: 329,226
Trainable params: 329,226
Non-trainable params: 0
Total mult-adds (M): 10.54
==============================================================================
Input size (MB): 0.10
Forward/backward pass size (MB): 0.23
Params size (MB): 1.32
Estimated Total Size (MB): 1.65
==============================================================================
```

Listing A.3: Example of convolutional neural network: *CIFAR10_Conv_Net*

```
Batch size: 32
Input shape: (32, 3, 32, 32)

==============================================================================
Layer (type:depth-idx)                  Output Shape              Param #
==============================================================================
CIFAR10_Conv_Net                        --                        --
--Conv2d: 1-1                           [32, 6, 28, 28]           456
--MaxPool2d: 1-2                        [32, 6, 14, 14]           --
--Conv2d: 1-3                           [32, 16, 10, 10]          2,416
--MaxPool2d: 1-4                        [32, 16, 5, 5]            --
--Linear: 1-5                           [32, 120]                 48,120
--Linear: 1-6                           [32, 84]                  10,164
--Linear: 1-7                           [32, 10]                  850
==============================================================================
Total params: 62,006
Trainable params: 62,006
Non-trainable params: 0
Total mult-adds (M): 21.06
==============================================================================
Input size (MB): 0.39
Forward/backward pass size (MB): 1.67
Params size (MB): 0.25
Estimated Total Size (MB): 2.31
==============================================================================
```

References

[ACG$^+$16] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Oct 2016.

[App21] Apple Inc. Apple Worldwide Developers Conference 2021 (WWDC2021), 2021.
`https://developer.apple.com/videos/play/wwdc2021/101/`.

[BBG20] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy profiles and amplification by subsampling. *Journal of Privacy and Confidentiality*, 10(1), January 2020.

[BDMN05] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The sulq framework. In *24th ACM SIGMOD International Conference on Management of Data / Principles of Database Systems, Baltimore (PODS 2005)*, June 2005.

[CMS11] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12(null):1069–1109, July 2011.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, page 202–210, New York, NY, USA, 2003. Association for Computing Machinery.

[DN04] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 528–544, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[DR16] Cynthia Dwork and Guy N. Rothblum. Concentrated differential privacy, 2016.

[DRV10] Cynthia Dwork, Guy Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS '10)*, page 51–60, Las Vegas, NV, 23–26 October 2010. IEEE, IEEE.

[Eur16] European Parliament and Council of the European Union. Regulation (EU) 2016/679 (General Data Protection Regulation), 2016. `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679`.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[KLN$^+$20] Yilin Kang, Yong Liu, Ben Niu, Xinyi Tong, Likun Zhang, and Weiping Wang. Input perturbation: A new paradigm between central and local differential privacy, 2020.

[KM21] Moritz Kirschte and Esfandiar Mohammadi. DPHelmet: Pre-training on public data to boost differentially private machine learning. In *Student Conference Proceedings 2021*, pages 2, 24, 31, 38. Infinite Science Publishing, 2021.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[Mir17] Ilya Mironov. Rényi differential privacy. *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, Aug 2017.

[MM18] Sebastian Meiser and Esfandiar Mohammadi. Tight on budget? tight bounds for r-fold approximate differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 247–264, New York, NY, USA, 2018. Association for Computing Machinery.

[SMM19] David Sommer, Sebastian Meiser, and Esfandiar Mohammadi. Privacy loss classes: The central limit theorem in differential privacy. *Proceedings on Privacy Enhancing Technologies*, 2019:245–269, 04 2019.

[SS98] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. 1998.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2017.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[XDW20] Depeng Xu, Wei Du, and Xintao Wu. Removing disparate impact of differentially private stochastic gradient descent on model accuracy, 2020.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.