Private Data Analysis - Homework 3

Yanning Dai (212008)

October 30, 2024

1. **Composing the Gaussian mechanism:** Consider a version of the Lemma 6.5 that is specific to the Gaussian mechanism: show that for every function $f : \mathcal{X}^n \to \mathbb{R}$ with global sensitivity Δ, for every pair of neighboring datasets D, D', there is a randomized algorithm F with the form $F(z) = az + b + \mathcal{N}(0, \rho^2)$ for some a, b, ρ such that

- If $U \sim \mathcal{N}(0, \sigma^2)$ then $F(U) \sim A(D)$ and
- If $V \sim \mathcal{N}(\Delta, \sigma^2)$ then $F(V) \sim A(D')$,

where $A(D) = f(D) + Z$ where $Z \sim \mathcal{N}(0, \sigma^2)$.

Lemma 6.5: For every pair of random variables X, Y such that $X \approx_{\epsilon,\delta} Y$, there exists a randomized map F such that $F(U) \sim X$ and $F(V) \sim Y$.

Answer:

We define parameters a, b, and ρ^2 as follows:

- $a = \frac{f(D')-f(D)}{\Delta}$. Since f has global sensitivity Δ and D and D' are neighboring datasets, we have $|a| \leq 1$.
- $b = f(D)$.
- $\rho^2 = \sigma^2(1 - a^2)$, ensuring $\rho^2 \geq 0$ since $|a| \leq 1$.

1) If $U \sim \mathcal{N}(0, \sigma^2))$, the mean and variance of $F(U)$ is calculated as:

$$\mathbb{E}[F(U)] = a \cdot 0 + b = b = f(D).$$

$$\mathrm{Var}[F(U)] = a^2\sigma^2 + \rho^2 = a^2\sigma^2 + \sigma^2(1 - a^2) = \sigma^2.$$

Thus, $F(U) \sim \mathcal{N}(f(D), \sigma^2) = A(D)$.

2) If $V \sim \mathcal{N}(\Delta, \sigma^2))$, the mean and variance of $F(U)$ is calculated as:

$$\mathbb{E}[F(V)] = a\Delta + b = \frac{f(D') - f(D)}{\Delta} \cdot \Delta + f(D) = f(D').$$

$$\mathrm{Var}[F(V)] = \sigma^2.$$

Thus, $F(V) \sim \mathcal{N}(f(D'), \sigma^2) = A(D')$.

2. Consider the following Algorithm 1 and prove the following statements

Algorithm 1 Generalized Random Response

```
1: Input Dataset $D = \{x_1, \cdots, x_n\}$ where $x_i$ is an $m$-bit string in $\left\{-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}\right\}^m \cup \{0\}$, privacy parameter $\epsilon$.
2: for $i = 1, \cdots, n$ do
3:     Sample $j \in \{1, 2, \cdots, m\}$ uniformly at random
4:     if $x_i \neq 0$ then
5:         Randomize $j$-th bit of $x_i$, i.e., $x_{i,j}$ as follows:
               $$z_{i,j} = \begin{cases} c_\epsilon m x_{i,j} & \text{w.p. } \frac{e^\epsilon}{e^\epsilon+1}, \\ -c_\epsilon m x_{i,j} & \text{w.p. } \frac{1}{e^\epsilon+1}, \end{cases}$$
           where $c_\epsilon = \frac{e^\epsilon+1}{e^\epsilon-1}$.
6:     else
7:         Generate a uniform bit $z_{i,j} \in \{-c_\epsilon\sqrt{m}, c_\epsilon\sqrt{m}\}$.
8:     end if
9:     Return $z_i = (0, 0, \cdots, z_{i,j}, 0, \cdots, 0)$, where $z_{i,j}$ is the $j$-th position of $z$.
10: end for
```

1. The algorithm is ϵ-LDP.
2. For each $x_i \in \left\{-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}\right\}^m \cup \{0\}$, $\mathbb{E}(z_i) = x_i$.

1. Proof:

An algorithm $\mathcal{A}$ is ϵ-LDP if, for any inputs x and x', and any possible output z,

$$\frac{\Pr[\mathcal{A}(x) = z]}{\Pr[\mathcal{A}(x') = z]} \leq e^\epsilon.$$

Consider an input dataset D, where each element x_i is independently processed by the algorithm.

We consider two cases:

1) If both x_i and x_i' are non-zero, for a given index $j \in \{1, \ldots, m\}$, the output $z_{i,j}$ is:

$$z_{i,j} = \begin{cases} c_\epsilon m x_{i,j} & \text{w.p. } \frac{e^\epsilon}{e^\epsilon+1}, \\ -c_\epsilon m x_{i,j} & \text{w.p. } \frac{1}{e^\epsilon+1}, \end{cases}$$

where $c_\epsilon = \frac{e^\epsilon+1}{e^\epsilon-1}$.

The ratio of probabilities for different inputs x_i and x_i', where they differ at position j, is:

$$\frac{\Pr[z_{i,j} = c_\epsilon\sqrt{m} \mid x_i]}{\Pr[z_{i,j} = c_\epsilon\sqrt{m} \mid x_i']} = e^\epsilon,$$

Therefore, the algorithm satisfies ϵ-LDP.

2) If one of x_i or x_i' is zero, we suppose that $x_i = 0$ and $x_i' \neq 0$.

For $x_i = 0$, the output $z_{i,j}$ is chosen uniformly from $\{-c_\epsilon\sqrt{m}, c_\epsilon\sqrt{m}\}$ with probability $\frac{1}{2}$. The output probabilities for $x_i' \neq 0$ are:

$$\Pr[z_{i,j} = c_\epsilon\sqrt{m} \mid x_i'] = \frac{e^\epsilon}{e^\epsilon+1}, \quad \Pr[z_{i,j} = -c_\epsilon\sqrt{m} \mid x_i'] = \frac{1}{e^\epsilon+1}.$$

Therefore, for $z_{i,j} = c_\epsilon\sqrt{m}$:

$$\frac{\Pr[z_{i,j} = c_\epsilon\sqrt{m} \mid x_i = 0]}{\Pr[z_{i,j} = c_\epsilon\sqrt{m} \mid x_i']} = \frac{\frac{1}{2}}{\frac{e^\epsilon}{e^\epsilon+1}} = \frac{e^\epsilon+1}{2e^\epsilon} \leq e^\epsilon.$$

For $z_{i,j} = -c_\epsilon\sqrt{m}$:

$$\frac{\Pr[z_{i,j} = -c_\epsilon\sqrt{m} \mid x_i = 0]}{\Pr[z_{i,j} = -c_\epsilon\sqrt{m} \mid x_i']} = \frac{\frac{1}{2}}{\frac{1}{e^\epsilon+1}} = \frac{e^\epsilon + 1}{2} \leq e^\epsilon.$$

The ratio of probabilities is bounded by e^ϵ, hence the algorithm satisfies ϵ-LDP.

To conclude, the algorithm is -LDP.

2. Proof:

1) If $x_i \neq 0$, the output $z_{i,j}$ at a given index j is randomized as follows:

$$\mathbb{E}[z_{i,j} \mid x_{i,j}] = c_\epsilon m x_{i,j} \cdot \left(\frac{e^\epsilon}{e^\epsilon + 1} - \frac{1}{e^\epsilon + 1} \right) = m x_{i,j}.$$

Since j is chosen uniformly, we have:

$$\mathbb{E}[z_i] = x_i.$$

2) If $x_i = 0$, the expected value of $z_{i,j}$ is:

$$\mathbb{E}[z_{i,j} \mid x_i = 0] = \frac{1}{2} c_\epsilon \sqrt{m} + \frac{1}{2}(-c_\epsilon\sqrt{m}) = 0.$$

Thus, $\mathbb{E}[z_i] = 0 = x_i$.

To conclude, for each x_i, $\mathbb{E}[z_i] = x_i$.

3. **Optimal Gaussian Mechanism:** In the lecture 5, we provided several Gaussian mechanisms (such as Theorem 5.7, Theorem 5.9 and Theorem 5.18). Try to compare these three mechanisms. You can use simple queries such as the mean or the average. You can go through the reference [1] in Lecture 5, and you can use the source code of the optimal Gaussian mechanism

https://github.com/BorjaBalle/analytic-gaussian-mechanism

Report:

1. Problem Description

We are given a dataset $D = \{x_1, x_2, \ldots, x_n\}$ where $x_i \in \{0,1\}^d$, and we aim to answer the average query function $f(D) = \frac{1}{n}\sum_{i=1}^n x_i$ using three Gaussian mechanisms:

Mechanism 1: For a given function $f : X^n \to \mathbb{R}$, we define the Gaussian mechanism as

$$A(D) = f(D) + \mathcal{N}\left(0, \frac{32\Delta^2 \log\frac{2}{\delta}}{9\epsilon^2}\right).$$

Then for any $\epsilon \leq 1$ and $\delta > 0$, the Gaussian mechanism is (ϵ, δ)-DP.

Mechanism 2: For a given function $f : X^n \to \mathbb{R}$, we define the Gaussian mechanism as

$$A(D) = f(D) + \mathcal{N}\left(0, \frac{2\Delta^2 \log\frac{1.25}{\delta}}{\epsilon^2}\right).$$

Then for any $\epsilon \in (0,1)$ and $\delta > 0$, the Gaussian mechanism is (ϵ, δ)-DP.

Mechanism 3: Let $f : X^n \to \mathbb{R}^k$ be a function with global ℓ_2-norm sensitivity Δ. For any $\epsilon \geq 0$ and $\delta \leq 1$, the Gaussian mechanism $A(D) = f(D) + Z$ with $Z \sim \mathcal{N}(0, \sigma^2 I_k)$ is (ϵ, δ)-DP if and only if

$$\Phi\left(\frac{\Delta}{2\sigma} - \frac{\epsilon\sigma}{\Delta}\right) - e^{\epsilon}\Phi\left(-\frac{\Delta}{2\sigma} - \frac{\epsilon\sigma}{\Delta}\right) \leq \delta.$$

2. Experiment Design

(1) Dataset Construction

The experiment will test various combinations of the dataset size n, dimensionality d, privacy budget ϵ, and privacy breach δ. The parameters are chosen as follows:

- n is set to values: [10, 50, 100, 500, 1000];
- d is set to values: [1, 5, 10, 50, 100];
- ϵ is set to values: [0.05, 0.1, 0.5, 1, 2, 5];
- δ is set to values: [10^{-6}, 10^{-5}, 10^{-4}].

For each combination of n, d, ϵ, and δ, generate 10 datasets where each $x_i \in \{0,1\}^d$.

(2) Implementing the Mechanisms

For each dataset, calculate the query function $f(D) = \frac{1}{n}\sum_{i=1}^n x_i$, and then apply the three mechanisms to perturb the result. Perform 100 experiments for each mechanism to reduce the effect of randomness.

(3) Utility Analysis

For each combination of n, d, and ϵ, evaluate the performance of these mechanisms using the following metrics:

- Mean Squared Error (MSE):

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (\hat{f}(D)_i - f(D))^2$$

- Standard Deviation (SD):

$$\sigma = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (\hat{f}(D)_i - \overline{\hat{f}(D)})^2}$$

where $\overline{\hat{f}(D)}$ is the average perturbed result over multiple experiments.

3. Results and Discussions

In this section, we present and analyze the experimental results for the Mean Squared Error (MSE) and Standard Deviation (SD) metrics under different parameter settings. The four figures below illustrate the results of these experiments.

(1) Effect of Privacy Breach Parameter δ

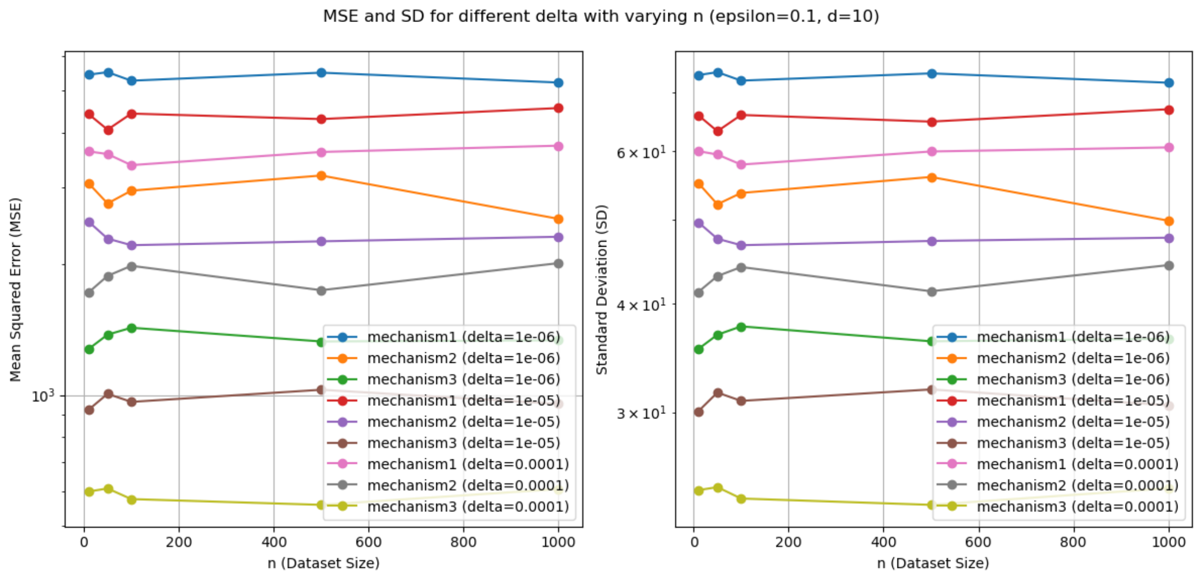


Figure 1: MSE and SD for different values of δ with varying dataset sizes n ($\epsilon = 0.1$, $d = 10$).

Figure 1 illustrates how MSE and SD change for different values of the privacy breach parameter δ as the dataset size n varies. We observe that for smaller values of δ, both MSE and SD are consistently higher, indicating a stronger privacy guarantee at the cost of increased error. Increasing δ reduces MSE and SD, suggesting a trade-off between privacy and accuracy. Mechanisms exhibit similar trends, but Mechanism 1 generally produces higher errors compared to Mechanisms 2 and 3.

(2) Effect of Dimensionality d

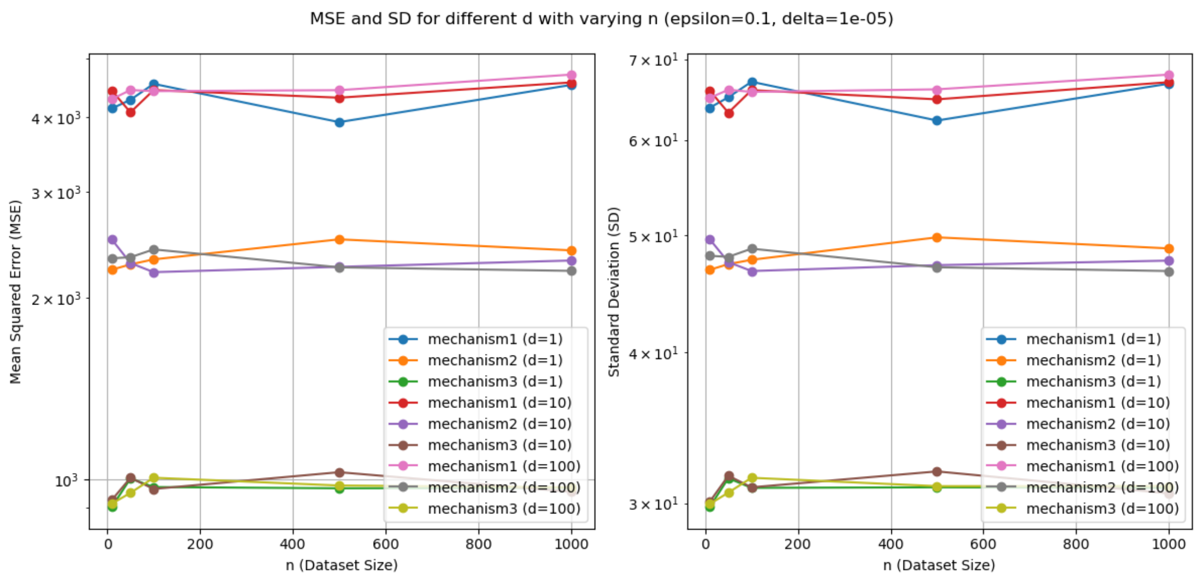


Figure 2: MSE and SD for different values of d with varying dataset sizes n ($\epsilon = 0.1$, $\delta = 10^{-5}$).

Figure 2 shows the effect of varying dimensionality (d) on MSE and SD with different dataset sizes. As d increases, the MSE and SD tend to increase for all mechanisms, particularly for Mechanism 1, which exhibits the highest sensitivity to increasing dimensionality. Mechanisms 2 and 3 show relatively stable performance, indicating their robustness to changes in the feature space dimensionality.

(3) Effect of Privacy Budget ϵ

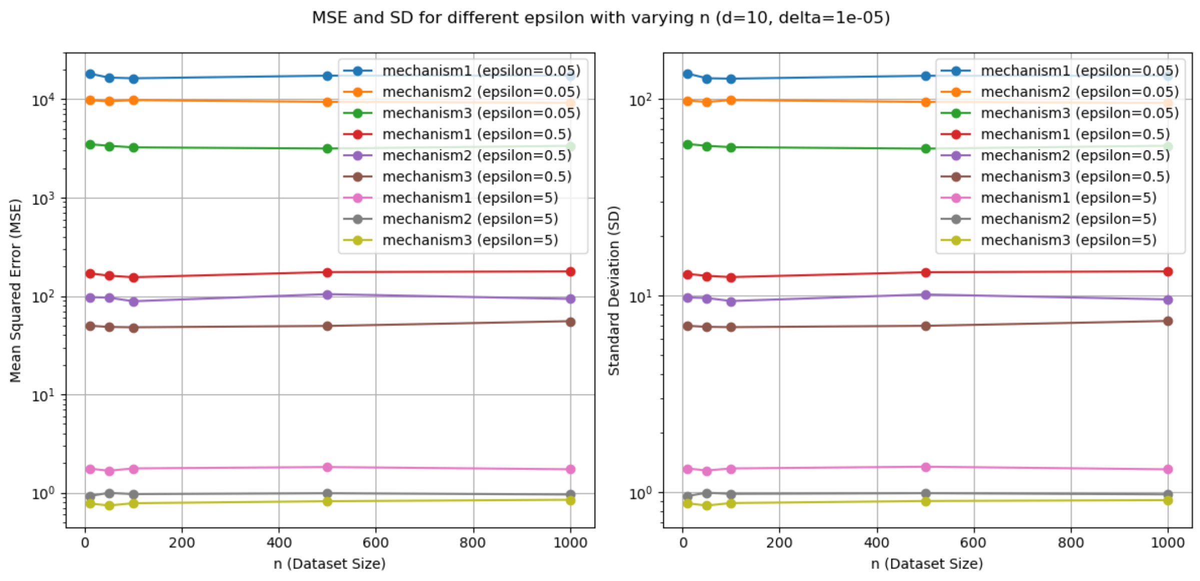


Figure 3: MSE and SD for different values of ϵ with varying dataset sizes n ($d = 10$, $\delta = 10^{-5}$).

Figure 3 shows the impact of the privacy budget (ϵ) on the MSE and SD across varying dataset sizes. Lower values of ϵ result in higher MSE and SD, indicating stronger privacy guarantees but increased error. As ϵ increases, the MSE and SD decrease, reflecting improved accuracy. Among the mechanisms, Mechanism 1 demonstrates the highest MSE for smaller ϵ, while Mechanisms 2 and 3 achieve better utility at lower privacy budgets.

(4) Effect of Dimensionality d

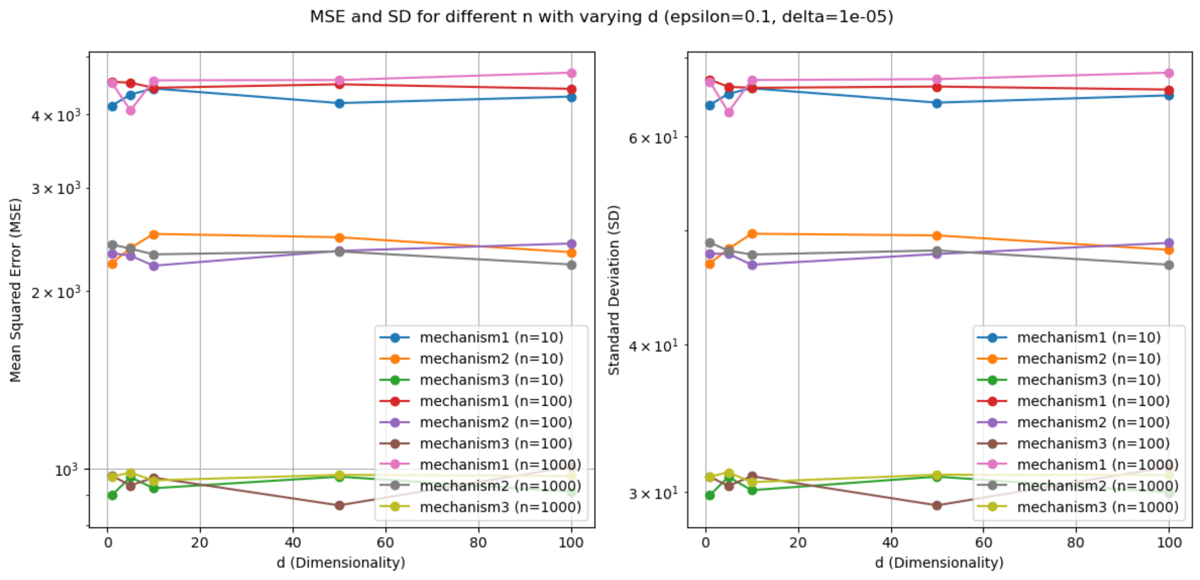


Figure 4: MSE and SD for different values of n with varying dimensionality d ($\epsilon = 0.1$, $\delta = 10^{-5}$).

Figure 4 highlights the effect of varying dimensionality (d) with different dataset sizes (n). As d increases, the MSE and SD remain relatively stable for larger datasets, indicating that the mechanisms are less sensitive to dimensionality when sufficient data is available. For smaller datasets, however, the increase in dimensionality results in a noticeable rise in both MSE and SD, suggesting that the data scarcity combined with high dimensionality negatively impacts utility.

(5) Comparasion of Mechanism Performance

Across the experiments, we observe that Mechanism 1 generally has higher MSE and SD compared to Mechanisms 2 and 3, which perform better in terms of accuracy and stability. Its higher error can be attributed to its more conservative noise addition, which ensures stronger privacy guarantees but at the expense of accuracy. Mechanism 3 consistently demonstrates the lowest error, especially as the dimensionality increases or the privacy parameters are relaxed, indicating that it offers a good balance between privacy and utility. Mechanism 2 also shows robust performance but tends to slightly lag behind Mechanism 3 in terms of error reduction.

Overall, the results indicate that Mechanism 3 is the most suitable for applications requiring a good trade-off between privacy and accuracy, particularly in higher-dimensional settings.

4. In this question, you will learn how to use the Opacus library to implement the DP-SGD algorithm for some deep learning tasks in PyTorch. You can find a tutorial at https://opacus.ai/.

You can select a training dataset by yourself, but a classification task is preferred. Here, you can investigate how different hyperparameters, such as the epoch, clipping threshold, and batch size, will affect the performance. Write a report for your experimental setting and results.

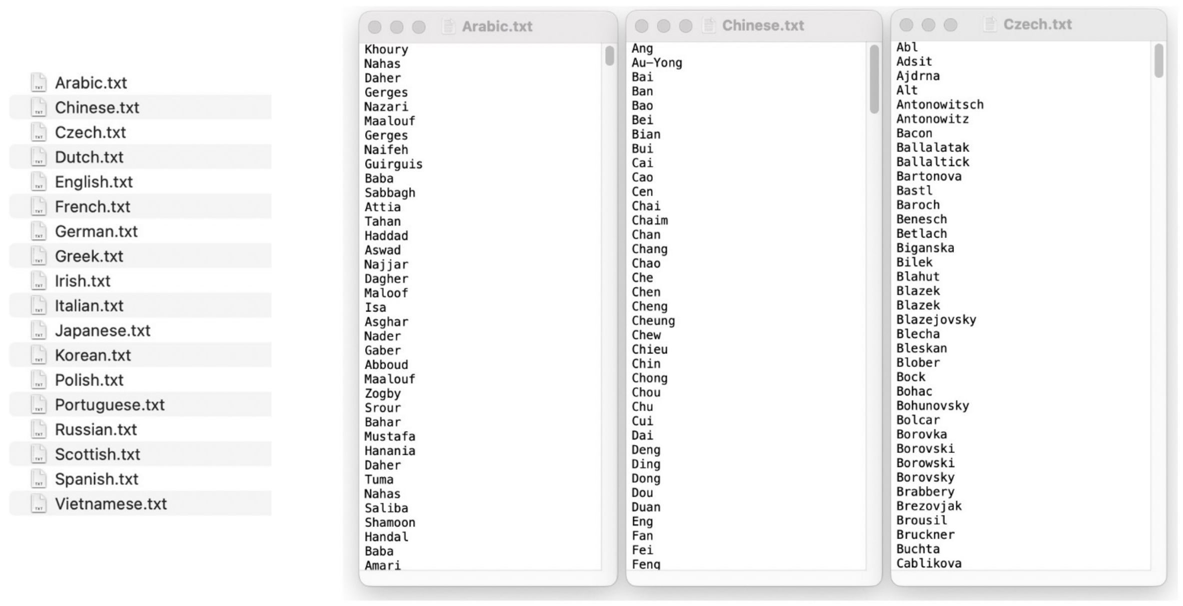
LSTM-Based Name Classifier with Differential Privacy

1. Task Description

The objective of this experiment is to build a character-level Long Short-Term Memory (LSTM)-based name classifier with differential privacy. The task is to classify names by their respective languages, which can provide insights into the relationship between names and their cultural origins. This is achieved by training the model with and without privacy constraints, allowing for a comparison of performance in both settings.

2. Dataset

The dataset used in this experiment is sourced from https://download.pytorch.org/tutorial/data.zip and contains name data across 18 languages [2]. Each name is represented as a string input, with the label corresponding to its respective language, as shown in Figure 5. This is a classification task where the model aims to predict the language category of a given name. Each name is converted into a byte sequence suitable for input into the LSTM model.



```
Arabic.txt
Chinese.txt
Czech.txt
Dutch.txt
English.txt
French.txt
German.txt
Greek.txt
Irish.txt
Italian.txt
Japanese.txt
Korean.txt
Polish.txt
Portuguese.txt
Russian.txt
Scottish.txt
Spanish.txt
Vietnamese.txt
```

Arabic.txt

```
Khoury
Nahas
Daher
Gerges
Nazari
Maalouf
Gerges
Naifeh
Guirguis
Baba
Sabbagh
Attia
Tahan
Haddad
Aswad
Najjar
Dagher
Maloof
Isa
Asghar
Nader
Gaber
Abboud
Maalouf
Zogby
Srour
Bahar
Mustafa
Hanania
Daher
Tuma
Nahas
Saliba
Shamoon
Handal
Baba
Amari
```

Chinese.txt

```
Ang
Au-Yong
Bai
Ban
Bao
Bei
Bian
Bui
Cai
Cao
Cen
Chai
Chaim
Chan
Chang
Chao
Che
Chen
Cheng
Cheung
Chew
Chieu
Chin
Chong
Chou
Chu
Cui
Dai
Deng
Ding
Dong
Dou
Duan
Eng
Fan
Fei
Feng
```

Czech.txt

```
Abl
Adsit
Ajdrna
Alt
Antonowitsch
Antonowitz
Bacon
Ballalatak
Ballaltick
Bartonova
Bastl
Baroch
Benesch
Betlach
Biganska
Bilek
Blahut
Blazek
Blazek
Blazejovsky
Blecha
Bleskan
Blober
Bock
Bohac
Bohunovsky
Bolcar
Borovka
Borovski
Borowski
Borovsky
Brabbery
Brezovjak
Brousil
Bruckner
Buchta
Cablikova
```

Figure 5: Structure of the Name Dataset

3. Model Architecture

The constrcuted model is based on an LSTM architecture, which is well-suited for sequential data like text. The model includes the following components:

- Embedding Layer: Maps each character (represented as a byte) to a dense vector of size 64.
- LSTM Layer: Processes the embedded sequences with a hidden size of 128 and a single layer of LSTM units.
- Output Layer: A fully connected linear layer that maps the LSTM's output to a vector of size

equal to the number of languages (classification output).

The vocabulary size is set to 259, comprising 256 standard UTF-8 characters plus three special tokens:

- `<s>` (start of sequence) with an index of 256,
- `</s>` (end of sequence) with an index of 257,
- `<pad>` (padding token) with an index of 258, used to pad sequences for uniform length within each batch.

During training, sequences within each batch are dynamically padded to match the length of the longest sequence in that batch. This padding is done using the `<pad>` token, ensuring input consistency and efficient computation.

4. Experimental Setup

(1) Parameters Setting

In this experiment, the WandB sweep functionality was used to automatically explore different hyperparameter combinations to find the optimal configuration. Below are the ranges of hyperparameters explored and the fixed parameters used in the experiment:

Hyperparameter Ranges:

- epochs: [20, 40, 60] — Number of training epochs.
- batch_size: [128, 256, 1024] — Different batch sizes.
- clipping_threshold: [0.5, 1.0, 2.0] — Gradient clipping threshold for differential privacy.

Fixed Hyperparameters:

- learning_rate: 2.0 — Learning rate.
- delta: 8×10^{-5} — Delta parameter in differential privacy.
- embedding_size: 64 — Dimension of the embedding layer.
- hidden_size: 128 — Dimension of the LSTM hidden layer.
- num_lstm_layers: 1 — Number of LSTM layers.

(2) Experimental Procedure

1. Data Preparation: The dataset was downloaded and processed by converting each name to a sequence of byte-level tokens. The dataset was split into 80% for training and 20% for testing.

2. Model Training with Differential Privacy: Using Opacus [1], differential privacy was applied to the training process. Privacy parameters, including clipping thresholds, were configured to ensure that each training example had limited influence on the model. During each epoch, the model was trained with the specified batch size and clipping threshold. WandB was used to log accuracy, loss, and privacy metrics (epsilon values).

3. Model Training without Differential Privacy: A second set of training was performed without differential privacy to serve as a baseline for comparison. The same batch sizes and epochs were used for consistency.

4. Evaluation: After each epoch, the model was evaluated on the test set to record accuracy and loss. At the end of both private and non-private training, the final test metrics were collected.

5. Results and Discussion

In this section, we analyze the effects of three main hyperparameters: epochs, batch size, and clipping threshold. Additionally, we compare model performance when using differential privacy (DP) versus non-private training.

(1) Effect of Epochs

Table 1: Results for Epochs Variation

Epochs	Batch Size	Clip Threshold	Accuracy	Epsilon
20	128	0.5	0.699	11.992
40	128	0.5	0.690	11.996
60	128	0.5	0.674	11.993

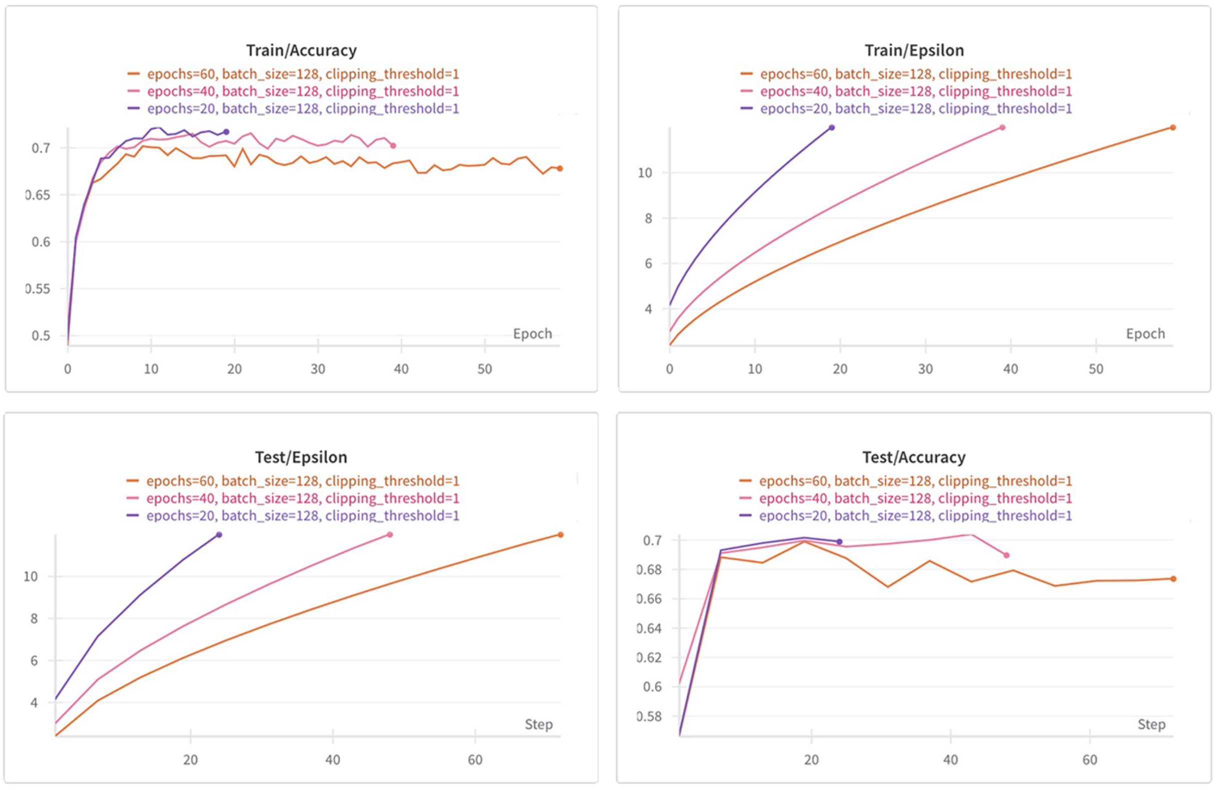


Figure 6: Results for Epochs Variation

In differential privacy settings, the number of epochs affects both accuracy and privacy loss. As shown in Figure 6, with each epoch, the model has more chances to learn, but added noise can distort gradients over prolonged training, especially as more epochs accumulate. This noise can prevent further accuracy gains, as seen when accuracy plateaus or even decreases after a certain point.

Despite more epochs, epsilon remains nearly constant due to controlled privacy budget allocation, which limits the cumulative privacy cost per epoch. This stability shows that while additional epochs might typically improve accuracy, in private training, more epochs may not always yield better results due to the compounded noise, highlighting the need for careful tuning.

(2) Effect of Batch Size

The batch size directly affects model stability, convergence, and privacy efficiency. In Figure 7, smaller batch sizes (e.g., 128) show steady improvements in accuracy while maintaining lower epsilon values. Larger batch sizes, like 1024, converge faster initially but result in lower accuracy,

Table 2: Results for Batch Size Variation

Epochs	Batch Size	Clip Threshold	Accuracy	Epsilon
60	128	0.5	0.718	11.993
60	256	0.5	0.747	11.994
60	1024	0.5	0.650	11.999

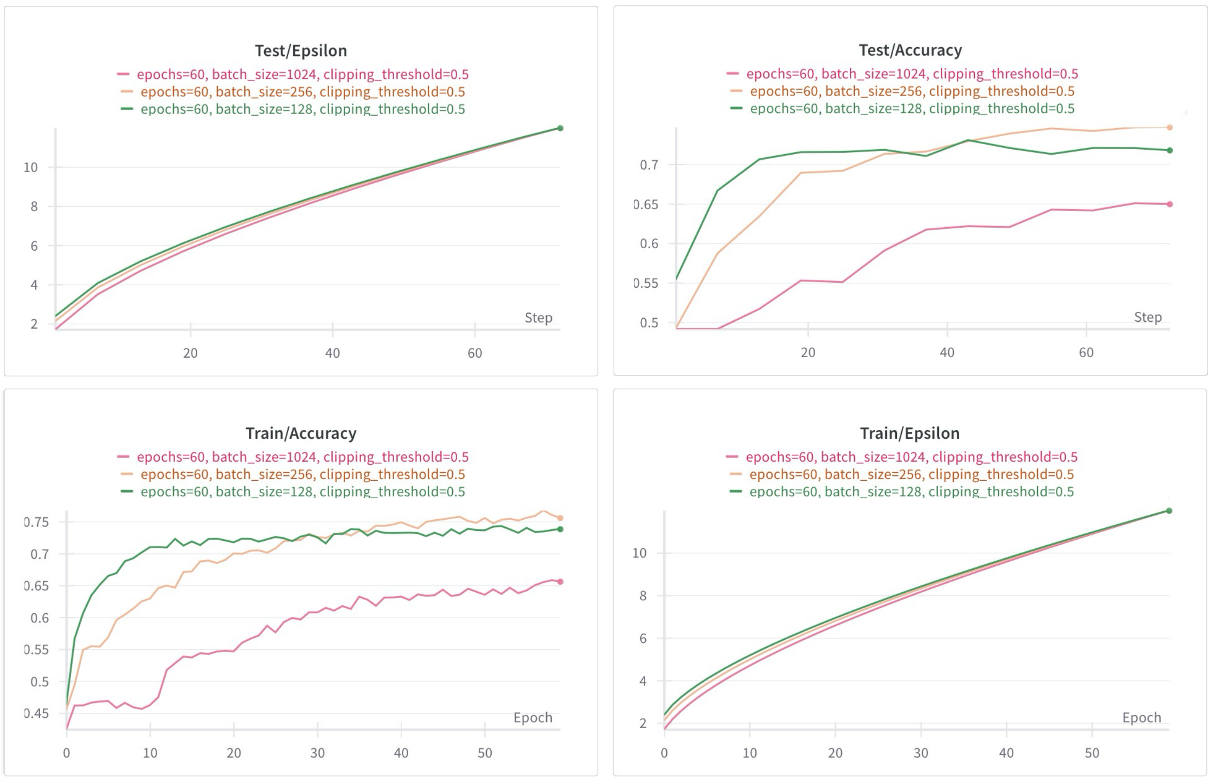


Figure 7: Results for Batch Size Variation

as seen in Table 2. Additionally, smaller batches require less noise to preserve privacy, which helps keep epsilon values manageable. The results illustrate that smaller batch sizes can yield better accuracy without a significant privacy trade-off.

(3) Effect of Clipping Threshold

Table 3: Results for Clip Threshold Variation

Epochs	Batch Size	Clip Threshold	Accuracy	Epsilon
60	128	0.5	0.718	11.993
60	128	1	0.674	11.993
60	128	2	0.661	11.993

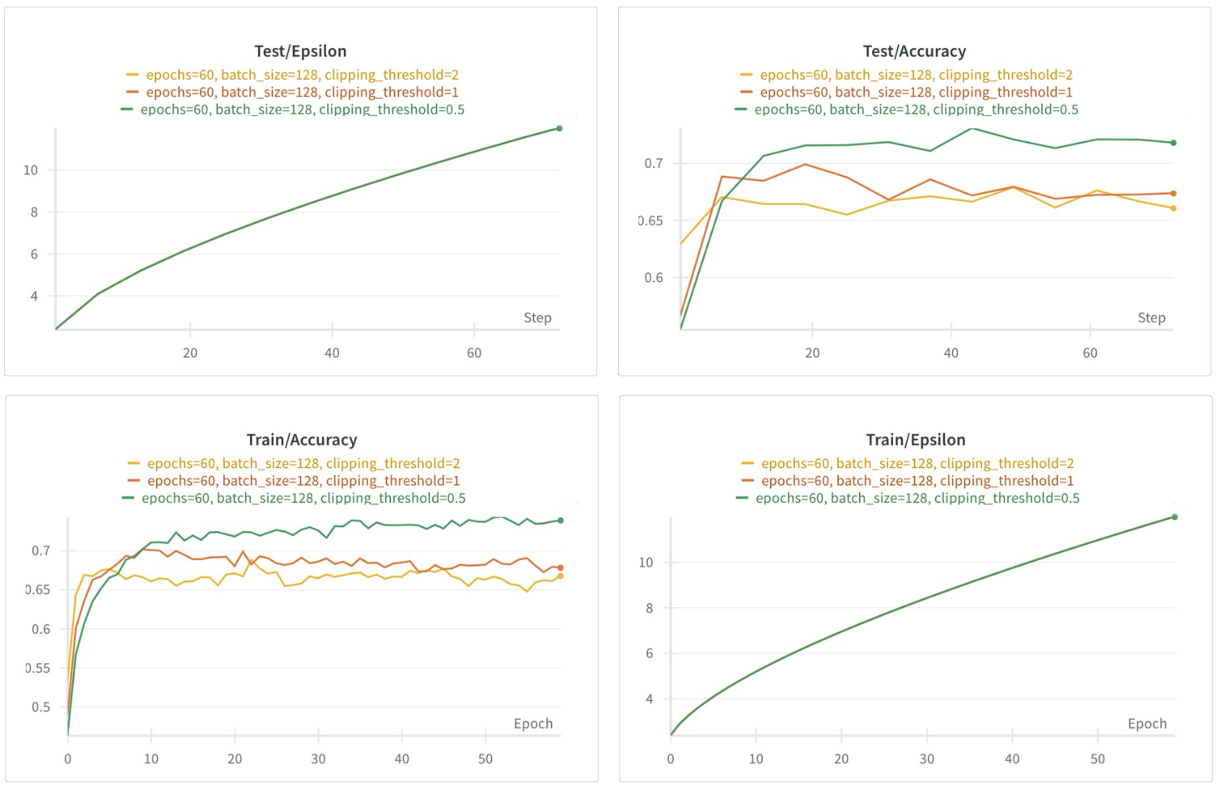


Figure 8: Results for Clip Threshold Variation

The gradient clipping threshold is crucial in differential privacy to control the maximum allowable gradient size, limiting the impact of individual samples. From Figure 9, lower clipping thresholds (0.5) show more stable improvements in accuracy, while higher thresholds allow for faster but less stable training. Table 3 reveals that a moderate threshold (e.g., 0.5) achieves a balance between accuracy (around 0.718) and privacy, while higher values lead to greater privacy loss without substantial accuracy gains.

(4) Privacy vs. Non-Privacy Comparison

We conducted a comparative analysis between DP and non-private training (see Figure 9). Below are the main results from both training types:

The comparison between private (differential privacy) and non-private training shows a clear trade-off between accuracy and privacy. The non-private model achieves a higher final accuracy of around 80.24%, whereas models trained with differential privacy reach approximately 70% accuracy. This difference is primarily due to the noise added in DP training, which protects privacy but can slightly impair the model's ability to capture fine-grained patterns in the data, thus limiting its peak performance [3].

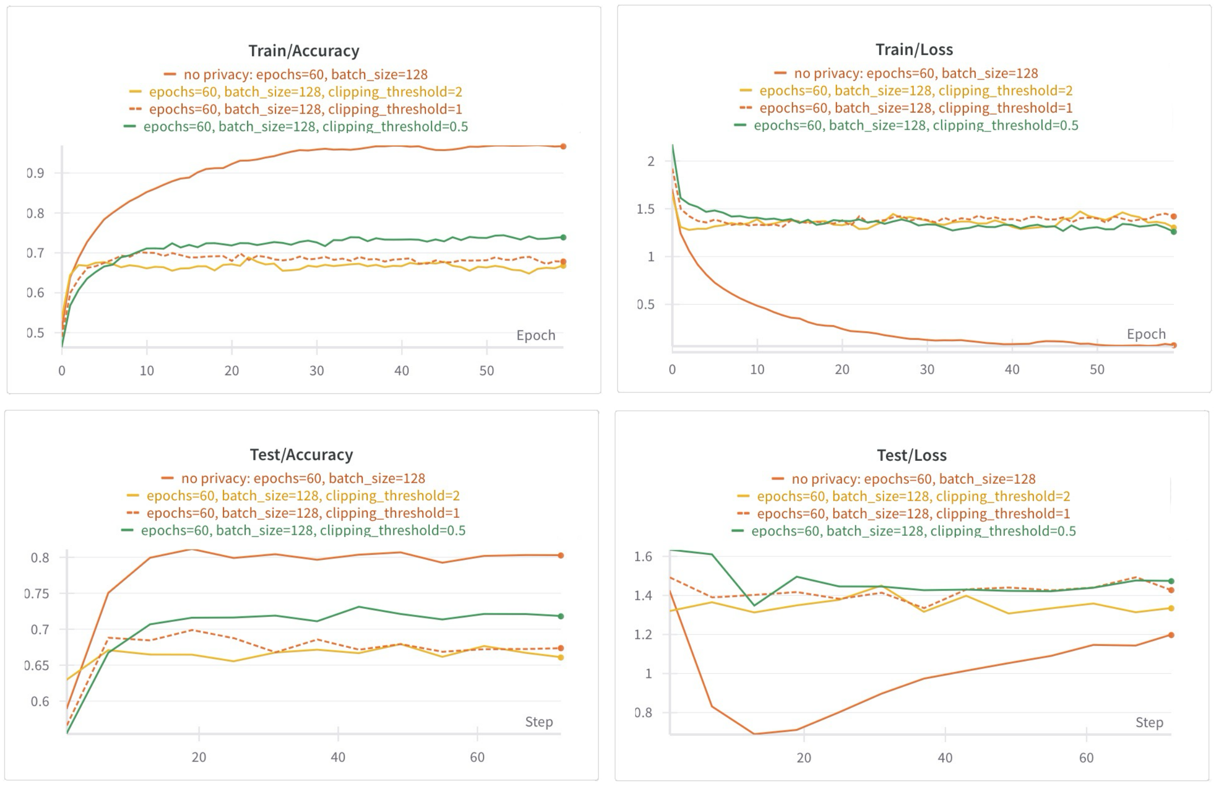


Figure 9: Results for Clip Threshold Variation

Both the DP and non-private models demonstrate good stability in terms of training and test loss, indicating that DP noise does not introduce instability in convergence. Instead, the noise subtly affects accuracy without disrupting the overall learning process. This result suggests that while DP training effectively preserves model stability, it comes with a slight accuracy reduction—a manageable trade-off depending on the privacy requirements. Lower clipping thresholds can help achieve a balanced performance, but some accuracy compromise remains inherent in DP settings.

6. Conclusion

This experiment demonstrates how differential privacy impacts model performance in text classification tasks. Increasing the number of epochs allows the model to learn more, but the noise added to ensure privacy can distort gradients over time, often leading to a plateau or even a decline in accuracy as training continues. Batch size also plays a crucial role: smaller batches tend to improve accuracy more steadily and require less noise, thereby maintaining tighter privacy control. Conversely, larger batch sizes stabilize training and speed up convergence but can lead to lower final accuracy and higher privacy costs. The clipping threshold, which limits gradient sizes, influences both learning efficiency and privacy. Lower thresholds generally stabilize accuracy improvements, while higher thresholds allow faster but less stable learning, often with increased privacy loss.

Overall, these findings emphasize the need to balance training duration, batch size, and clipping thresholds in DP settings to achieve optimal accuracy without compromising privacy.

References

[1] Opacus: A Library for Differential Privacy in PyTorch. Available at: https://opacus.ai/. Accessed on 31st Otc, 2024.

[2] PyTorch Char-RNN Classification Tutorial. Available at: https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html. Accessed on 31st Otc, 2024.

[3] Abadi, Martin, et al. ”Deep learning with differential privacy.” Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016.