Differential Privacy in Federated Learning on MIMIC-IV: A Study of Heterogeneous Healthcare Clients

Shrawani Gulhane Gaurav Mehta Vasudev Menon

April 29, 2025

1 Background and Introduction

Industries such as healthcare face a pressing need to develop machine learning frameworks that ensure both high accuracy and privacy. Traditional machine learning requires aggregating raw data on a central server, which poses significant risks of data breaches, unauthorized access, and regulatory violations, especially when dealing with personally identifiable information (PII) such as sensitive medical records. Federated Learning (FL) [14] has emerged as a popular alternative, allowing model training to occur locally on client devices or within institutions, thus keeping data decentralized, which reduces risks of data misuse and maintaining patient privacy.

However, privacy concerns in healthcare machine learning persist even when using de-identified datasets. Although de-identification is often viewed as sufficient under legal standards like the Health Insurance Portability and Accountability Act (HIPAA), recent research has demonstrated that such protections are not foolproof. In fact, a recent publication introduced the MVA (Minimum Viable Attack), a re-identification method that showed how records from HIPAA-compliant de-identified datasets could still be linked back to individuals using a small number of identifiers such as age, sex, and hospital visit information [8]. This work highlights the fact that de-identification does not itself eliminate privacy risk com-

pletely, and this risk is especially exacerbated when attackers possess additional, auxiliary information. Due to these reasons, additional layers of protection are needed when working with sensitive health data, regardless of whether or not it is sufficiently de-identified.

To address these limitations, our project investigates the application of Distributed Differential Privacy (DDP) in FL systems with heterogeneous client data. DDP utilizes the concept differential privacy by allowing each client to add calibrated noise to its local updates before transmission, thereby enhancing privacy while preserving the benefits of federated computation. Our project aims to understand how the incorporation of DDP affects model performance under varying privacy budgets, and to quantify the trade-offs between accuracy and privacy in the presence of skewed client distributions.

We implement this system using the Flower federated learning framework [2], which offers modularity and compatibility with PyTorch [18]. Flower enables simulation of multiple clients and seamless integration of privacy-preserving techniques via its modular privacy APIs. Compared to alternatives such as NVFlare and PySyft [21, 23], Flower requires minimal setup and can run out-the-box on platforms such as Google Colab. The clients in our experiments represent demographic splits within a medical dataset, specifically racial groups derived from the MIMIC-IV electronic health records [9, 10]. Each client trains a local model on a subset of the data and adds Gaussian noise to the parameters before sharing them with the server for aggregation via federated averaging (FedAvg).

With this setup, we explored some key questions, such as: How does DDP impact model convergence and accuracy? What is the relationship between the privacy budget ϵ and performance? By working with structured medical data and dividing it by demographic groups, we aim to offer a practical look at how federated learning can be used in real healthcare settings. This project helps connect the theoretical ideas behind differential privacy with the challenges of applying them in real-world systems that involve diverse and uneven data.

2 Data Sets

The primary dataset used in this project is MIMIC-IV, a publicly available collection of de-identified health records from ICU stays at Beth Israel Deaconess Medical Center [9,

10], which we accessed through the PhysioNet repository [4]. MIMIC-IV is one of the most comprehensive healthcare datasets available, spanning over 100GB of structured and unstructured clinical information distributed across dozens of CSV files. Due to the enormous size and complexity of the dataset, one of the primary challenges we faced was efficiently accessing and processing the data within our limited computational and storage environment.

Access to the MIMIC-IV dataset requires completing a series of steps and training courses due to the sensitive nature of the health data involved. As part of the access process, the following steps were completed: (1) created PhysioNet credentialed user accounts, (2) completed the "Data or Specimens Only Research" training module via the CITI Program, and (3) signed and submitted a Data Use Agreement (DUA) for the project. Only after fulfilling these requirements were we granted access to download and utilize the MIMIC-IV data. This controlled access process ensures ethical and responsible use of patient information, even though the data is in a de-identified format.

Our initial attempt involved using `pandas` [13] to load the dataset in chunks and extract relevant information. However, due to hardware limitations, this approach proved infeasible. We then loaded the dataset into a SQL database, but encountered performance bottlenecks such as SQL's constraint on query execution times, which often exceeded the 30 second limit for complex fetch operations. To move forward, we resorted to manually retrieving data using Power Query. Although this process was highly inefficient, taking nearly six hours to complete, it enabled us to prepare an intermediate dataset for modeling, which was used with `pandas` to format the data. This final solution allowed us to retrieve some data, which was then structured for data modeling and analysis.

From the MIMIC-IV data repository, we focused on three core CSV files:

- `admissions.csv`: Contains hospital admission data, including race and mortality.
- `chartevents.csv`: Time-stamped records of patient vital signs and bedside observations.
- `d_items.csv`: Provides human-readable labels for item IDs used in `chartevents.csv`.

We extracted a set of clinically relevant variables: systolic and diastolic arterial blood pressure (ABP), glucose level, heart rate, respiratory rate, and body temperature. For each

patient, we aggregated time-series data by computing the mean value per measurement type. After merging with demographic data, we dropped any patient records that contained missing values in any of the selected features to ensure consistency and reliability of the input to our models.

```
~91% of the data has Hospital_Expire_Flag = 0
~9% of the data has Hospital_Expire_Flag = 1

Balanced data:
~50% of the data has Hospital_Expire_Flag = 0
~50% of the data has Hospital_Expire_Flag = 1
      subject_id  ABP Diastolic  ABP Systolic  Glucose  Heart Rate  \
0       16749381          55.45        118.26   135.20       88.56
1       11929406          50.38        114.92   184.00       97.73
2       11726221          48.19        116.82   163.50       85.58
3       12626863          59.44        102.93   167.92      101.50
4       17264263          63.23        114.59   119.00       89.52
...          ...            ...           ...      ...         ...
4721    18690354          54.35         98.98   221.60      108.32
4722    13390681          64.48        120.39   148.11       79.26
4723    13197784          61.93        146.90   172.56       91.24
4724    15947104          79.54        142.47   152.21       91.00
4725    15218695          67.77        128.35   115.14       77.18

      Respiratory Rate  Temperature     race  Hospital_Expire_Flag
0                14.21        98.34    WHITE                     1
1                27.08        98.31  UNKNOWN                     1
2                18.45        98.08    WHITE                     0
3                22.73        97.97    WHITE                     1
4                20.48        98.40    WHITE                     0
...                ...          ...      ...                   ...
4721             21.72        97.64  UNKNOWN                     1
4722             15.18        98.95    WHITE                     0
4723             15.02        97.95    WHITE                     1
4724             21.11        99.03  UNKNOWN                     1
4725             16.09        98.20    WHITE                     0

[4726 rows x 9 columns]
```

Figure 1: Overview of the balanced patient dataset. Each row corresponds to a unique patient identified by `subject_id`. Clinical features include ABP Diastolic, ABP Systolic, Glucose level, Heart Rate, Respiratory Rate, and Temperature. The demographic feature included is `race`. The target variable for prediction is `Hospital_Expire_Flag`.

The target variable `Hospital_Expire_Flag` was assigned a value of 1 if the patient died in any hospital admission, and 0 otherwise. This allowed us to define a binary mortality prediction task. As shown in Figure 1, the raw label distribution was highly imbalanced, with only around 9% of records representing expired patients. To correct for this and stabilize

training, we applied random undersampling to the majority class (unexpired patients) and created a balanced dataset with 4,704 patient records, as shown in Figure 2.

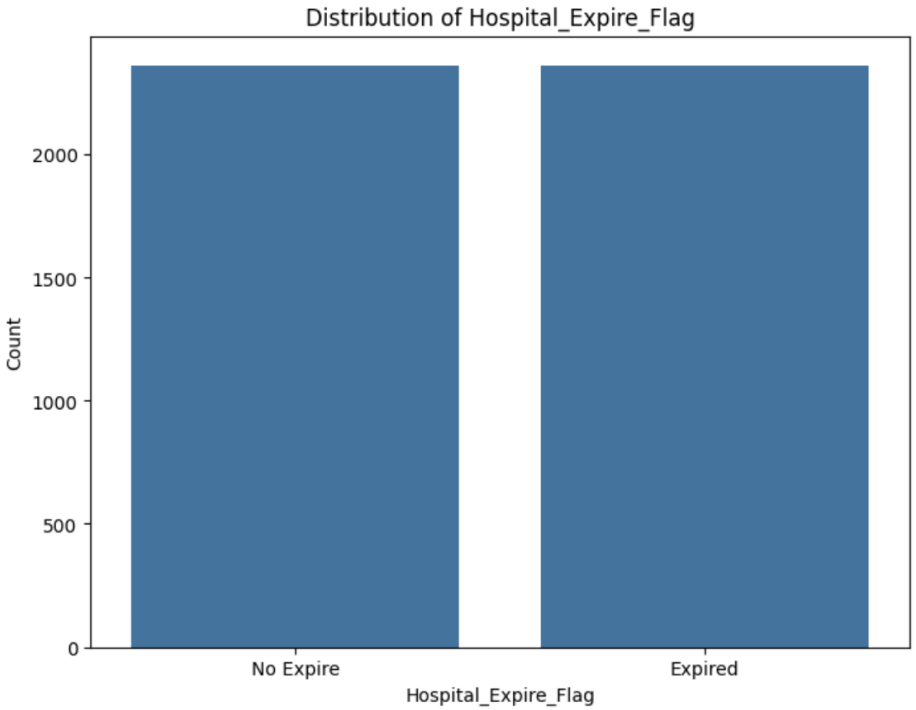

Figure 2: Balanced Outcome Distribution of `Hospital_Expire_Flag` After Preprocessing

To prepare the data for training, we normalized all continuous features using standard scaling via `scaler.fit_transform()`. This ensured that all input features contributed equally to the parameter computations and improved model convergence.

Mathematically, standard scaling transforms each feature such that its distribution has a mean of 0 and a standard deviation of 1. For a given feature x, this is done using the following formula:

$$x' = \frac{x - \mu}{\sigma}$$

where μ is the mean of the feature values across the dataset and σ is the standard deviation. The `fit_transform()` method from `sklearn.preprocessing.StandardScaler` computes these statistics and applies the transformation to all values [19]. This standardization ensures that features with larger numeric ranges do not dominate model updates, leading to faster and more stable convergence.

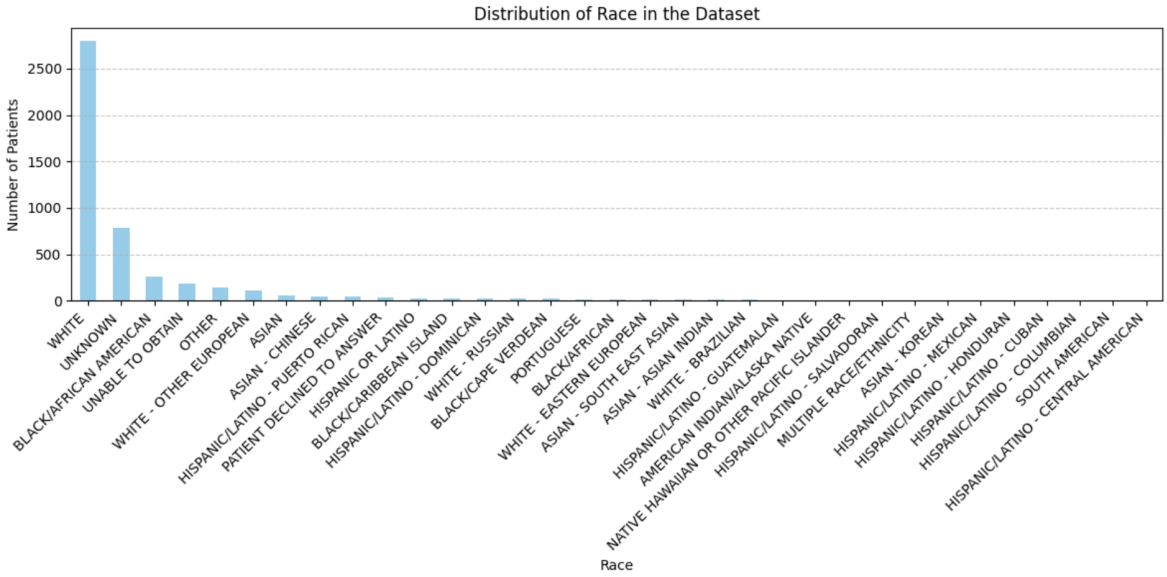

Figure 3: Distribution of Detailed Race Labels in the Dataset

Next, we assigned each patient to a Federated Learning (FL) client using their recorded race. Figure 3 illustrates the original race distribution across more than 30 descriptors. To improve consistency we grouped these into broader categories: `WHITE`, `BLACK`, `ASIAN`, `HISPANIC/LATINO`, `NATIVE`, `OTHER`, and `UNKNOWN`. Each FL client corresponds to one of these race groups, such that all patients assigned to a given client belong to the same race category. Figure 4 shows the overall distribution of patients across these grouped categories.

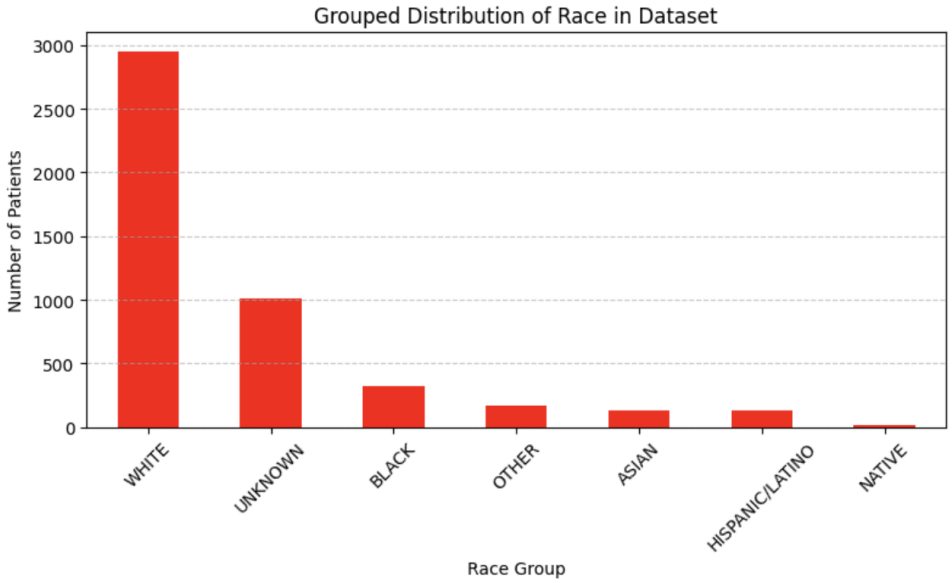

Figure 4: Grouped Race Distribution Used to Define Federated Clients

Unlike the label classes, we did not balance racial group sizes across clients. This decision was made model the demographic skew found in real-world clinical datasets. Each race group is treated as a distinct client, introducing client specific variations in both data volume and distribution. This allowed us to use heterogeneous clients for our FL experiments.

To visualize the range and behavior of clinical features, Figure 5 presents histograms for the six selected variables. The plots reveal patterns in real patient data, including slightly skewed glucose levels and normally distributed temperatures.

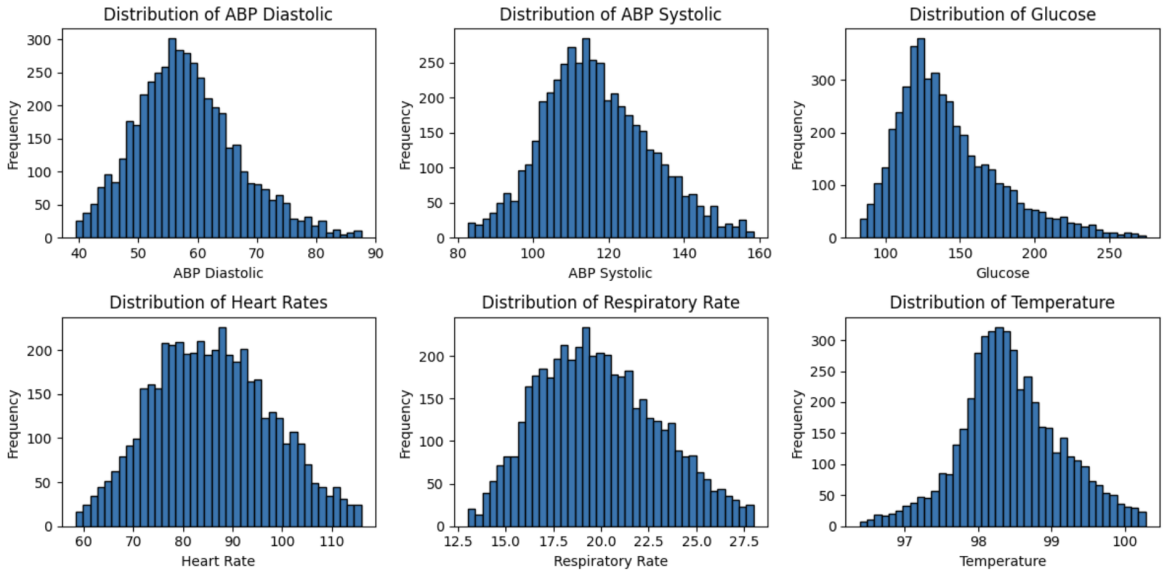

Figure 5: Distribution of Clinical Features Across All Patients

We denote the dataset by $\mathcal{D} = \bigcup_{i=1}^{K} \mathcal{D}_i$, where each client i receives a dataset $\mathcal{D}_i = \{(x_j, y_j)\}_{j=1}^{n_i}$ composed of feature vectors x_j and labels y_j. Here, $K = 7$ (`WHITE`, `BLACK`, `ASIAN`, `HISPANIC/LATINO`, `NATIVE`, `OTHER`, `UNKNOWN`). This structure preserves real-world demographic distributions and ensures that we exhibit heterogeneity in our dataset.

3 Methods and Algorithms

Our system is implemented using the Flower federated learning framework, simulating decentralized training across heterogeneous clients using real clinical data. In this section, we first describe the client-side modeling approach, including model architecture and training logic, followed by the federated learning strategy.

Client-Side Model Training

Each federated client receives a private subset of patient records based on racial groupings. The local dataset $\mathcal{D}_i = \{(x_j, y_j)\}_{j=1}^{n_i}$ consists of normalized clinical features $x_j \in \mathbb{R}^d$ and binary labels $y_j \in \{0, 1\}$, where $d = 6$ (representing the 6 selected clinically relevant variables). The binary label indicates whether a patient died during any hospital admission (`Hospital_Expire_Flag`).

To model this binary classification task, we define a simple feedforward neural network, `MortalityNN`, using PyTorch. The model architecture is as follows:

$$
\begin{aligned}
h_1 &= \mathrm{ReLU}(W_1 x + b_1), & W_1 &\in \mathbb{R}^{4\times d},\\
h_2 &= \mathrm{ReLU}(W_2 h_1 + b_2), & W_2 &\in \mathbb{R}^{2\times 4},\\
\hat{y} &= \sigma(W_3 h_2 + b_3), & W_3 &\in \mathbb{R}^{1\times 2},
\end{aligned}
$$

where ReLU denotes the Rectified Linear Unit [16], defined as $\mathrm{ReLU} = \max(x, 0)$, and $\sigma(x)$ denotes the sigmoid activation function, defined as $\frac{1}{1+e^{-x}}$, producing a mortality probability $\hat{y} \in [0, 1]$.

Table 1: Architecture of `MortalityNN`

Layer	**Type**	**Output Size**	**Activation**
Input	Linear	4	ReLU
Hidden	Linear	2	ReLU
Output	Linear	1	Sigmoid

Training is performed locally on each client using the binary cross-entropy loss function:

$$\mathcal{L}_{\mathrm{BCE}} = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

which is typically used in binary classification tasks.

Optimization is done using the Adam optimizer [11], which combines the advantages of both AdaGrad and RMSProp by adapting the learning rate for each parameter using

estimates of first and second moments of the gradients. A default learning rate of 0.001 is used. Clients train for 10 local epochs per round using a batch size of 32. For evaluation, we compute the standard binary classification accuracy metric using each client's validation set.

This architecture was kept small and lightweight to ensure fast, consistent local training across all clients, even those with limited computational resources. Additionally, the focus of this project is on exploring federated learning and differential privacy, rather than our neural network model.

Federated Learning and Aggregation Strategy

Once local training is completed, each client sends its updated model parameters back to a central server. The server performs a weighted aggregation of client updates using the Federated Averaging (FedAvg) algorithm [14]. The global model parameters θ_{t+1} are updated as:

$$\theta_{t+1} = \sum_{i=1}^{K} \frac{n_i}{n} \theta_i$$

where:

- K is the number of clients participating in the round,
- n_i is the number of training samples at client i,
- $n = \sum_{i=1}^{K} n_i$ is the total number of samples,
- θ_i is the updated (noisy and clipped) model from client i.

To enhance privacy, we utilize client-side differential privacy using Flower's `LocalDpMod`. This module first applies ℓ_2-norm clipping to each client's model update. Then, it calculates a Gaussian noise value, which is added directly to the model parameters before they are sent to the server. This ensures that no raw weights are plainly transmitted. The calculations for this Gaussian noise value depends on a few privacy hyperparameters, as follows:

- Clipping norm: 1.0

- Sensitivity: 1.0
- $\delta = 10^{-5}$
- $\epsilon \in \{0.1, 0.5, 1, 2, 5, 10, 50, 100\}$

Here, ϵ is known as the privacy budget, dictates the amount of privacy added to the weights. A lower epsilon value, such as $\epsilon = 0.1$, adds more noise and provides stronger privacy, but may negatively affect model performance. Higher values, like $\epsilon = 50$ or 100, offer weaker privacy, but could allow the model obtain better accuracies. We vary the privacy budget ϵ across a wide range of values to evaluate the trade-off between model accuracy and privacy. The delta parameter $\delta = 10^{-5}$ sets the probability that the privacy guarantee fails, consistent with thresholds commonly used in the literature. The clipping norm constrains the ℓ_2-norm of each client's model update.

The amount of Gaussian noises σ added to each client's model update for the (ϵ, δ)-differential privacy is determined by:

$$\sigma \geq \frac{\sqrt{2\ln(1.25/\delta)} \cdot \Delta}{\epsilon}$$

where Δ is the sensitivity of the client update (enforced via clipping), ϵ is the privacy budget, and δ is the failure probability [3]. As ϵ varies, the required noise scale σ adjusts accordingly. Smaller ϵ values (e.g., 0.1) result in significantly larger noise magnitudes, resulting in higher privacy guarantees.

We utilize standard values for differential privacy hyperparameters, taken from prior work and literature. The clipping norm and sensitivity are both set to 1.0, following standard conventions [1]. The delta parameter is set to $\delta = 10^{-5}$, a commonly used threshold in differentially private machine learning that satisfies the requirement $\delta \ll 1/n$ for a dataset with multiple user samples [3, 17].

This implementation of distributed differential privacy ensures client privacy protection without altering their training process. The server aggregates the noisy, clipped updates into a new global model, which is then sent to clients for the next round.

Figure 6 [12] illustrates this process: each client trains locally, applies the `LocalDpMod`

to add noise to its model parameters, and sends these noisy updates to the central server. The server then aggregates the updates and broadcasts the new global model for the next round of training.

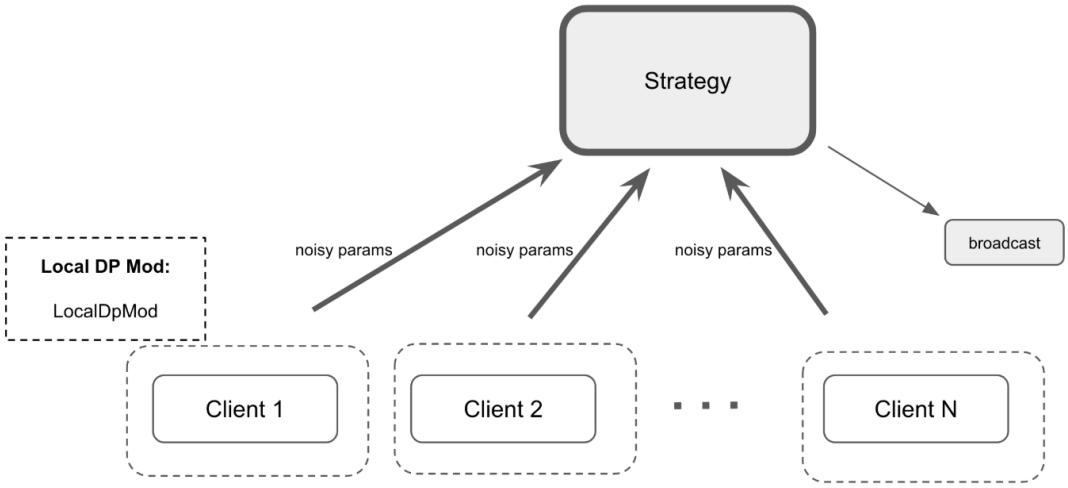

Figure 6: Federated learning with client-side local differential privacy via `LocalDpMod`. Noisy parameters are sent to the central server, which aggregates them and broadcasts the updated model.

The entire process is repeated over 25 global rounds, allowing the model to incrementally improve while preserving the privacy of each client's data. By simulating real-world demographic splits and applying DDP at the client level, we are able to evaluate model accuracy and privacy trade-offs in a heterogeneous federated environment.

4 Results

Accuracy Across Federated Rounds and Clients

Figure 7 shows the classification accuracy of each client over 25 global federated learning rounds without differential privacy being implemented. Each colored line represents one client, while the bold black line reflects the global average accuracy across all clients. We can see that the individual client accuracies vary significantly, but the average accuracy tends to slowly increase between the rounds, and plateaus around 0.70 by the final round.

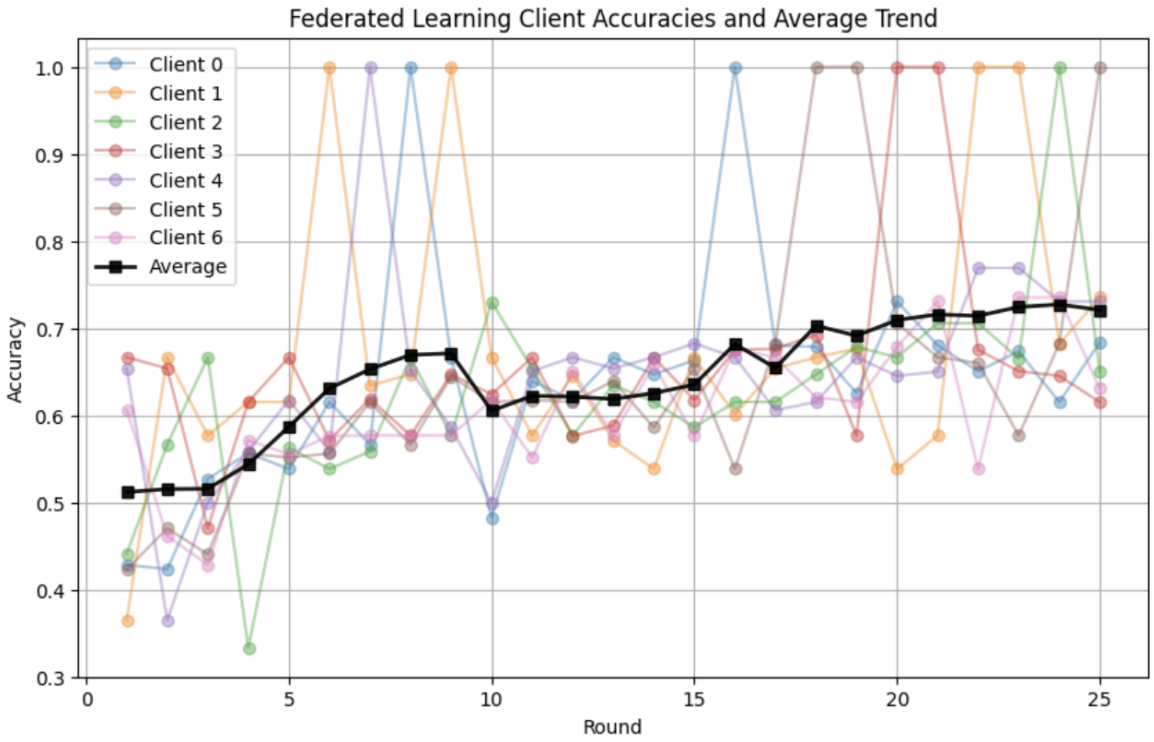

Figure 7: Client-specific accuracy and global average accuracy across 25 global rounds of federated training.

Effect of Differential Privacy Budget on Final Accuracy

To evaluate the trade-off between privacy and accuracy, we conducted a sweep over various values of the privacy budget $\epsilon \in \{0.1, 0.5, 1, 2, 5, 10, 50, 100\}$. Figure 8 presents the weighted average model accuracy at the final global round for each ϵ value.

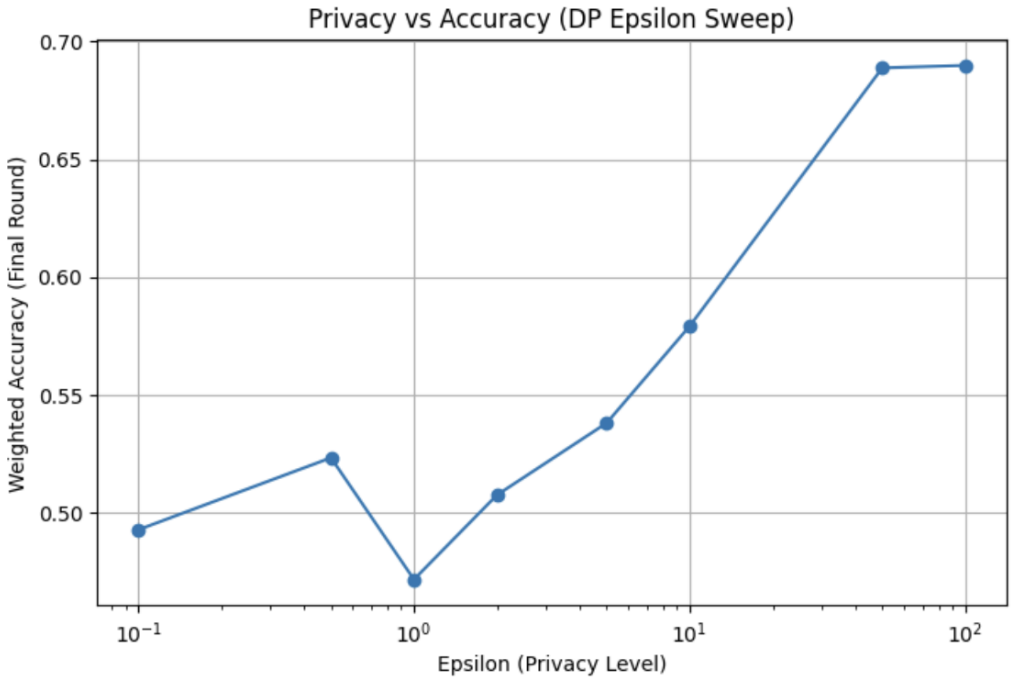

Figure 8: Weighted final-round accuracy as a function of privacy budget ϵ.

Because the ϵ values span several orders of magnitude, we used a logarithmic scale on the x-axis to better visualize the relationship between privacy level and model accuracy.

5 Discussion and Conclusions

The fluctuation in client-level accuracy is expected, as clients differ in both data size and distribution. There is rapid accuracy increase with occasional drops for some clients, which can be explained by the server returning aggregated models after each training round. Additionally, many of the clients that experience large fluctuations in accuracy between rounds are typically those that do not have a large number of samples, and the clients with lower fluctuations are those with the larger samples. In fact, the majority demographic class within our dataset, the `WHITE` group, is stored in Client 6, shown in the pink line, and this client does not experience large accuracy fluctuations like the others. Despite this, the overall trend is positive, with accuracy improving throughout rounds, which indicates a successful convergence of the federated model.

Additionally, the results from the privacy level and model accuracy plot demonstrate a

clear positive correlation between the privacy budget ϵ and final accuracy. When ϵ is small (e.g., 0.1 or 1), the added noise significantly hinders model performance, resulting in final accuracies near 0.49–0.52. As ϵ increases, the noise level decreases, allowing the model to converge more effectively. At $\epsilon = 50$ and above, the accuracy stabilizes near 0.69–0.70, close to non-private baseline levels. This confirms the expected privacy accuracy trade-off.

Overall, this work demonstrates the integration of Distributed Differential Privacy into Federated Learning systems applied to real-world ICU data. Our approach uses demographic based client partitioning to emulate realistic heterogeneity in a setting where data privacy is a foremost priority. Our findings indicate that while FL with DDP can maintain good performance, low accuracy and noise levels must be carefully balanced.

Future work can explore different client participation strategies and other commonly used aggregation strategies, such as FedAdam, FedAdagrad, FedYogi, and Bulyan [15, 20]. Moreover, the Flower framework also allows custom server aggregation strategies to be defined, which could be used to explore FL in the context of multimodal data spanning across numerous clients. Server-side privacy preserving algorithms could also be an area of future research. Ultimately, this project highlights the importance of tailoring privacy preserving techniques to real-world datasets.

References

[1] Martin Abadi et al. "Deep Learning with Differential Privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS'16. ACM, Oct. 2016. DOI: 10.1145/2976749.2978318. URL: http://dx.doi.org/10.1145/2976749.2978318.

[2] Daniel J. Beutel et al. *Flower: A Friendly Federated Learning Research Framework*. 2022. arXiv: 2007.14390 [cs.LG]. URL: https://arxiv.org/abs/2007.14390.

[3] Cynthia Dwork and Aaron Roth. "The algorithmic foundations of differential privacy". en. In: *Found. Trends Theor. Comput. Sci.* 9.3-4 (2013), pp. 211–407.

[4] A. L. Goldberger et al. "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals". In: *Circulation* 101.23 (2000). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215, e215–e220.

[5] Mehak Gupta et al. *An Extensive Data Processing Pipeline for MIMIC-IV*. 2022. arXiv: 2204.13841 [cs.LG]. URL: https://arxiv.org/abs/2204.13841.

[6] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[7] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[8] Victor Janmey and Peter L Elkin. "Re-identification risk in HIPAA DE-identified datasets: The MVA attack". en. In: *AMIA Annu. Symp. Proc.* 2018 (Dec. 2018), pp. 1329–1337.

[9] Alistair Johnson et al. *MIMIC-IV*. 2024.

[10] Alistair E W Johnson et al. "MIMIC-IV, a freely accessible electronic health record dataset". en. In: *Sci. Data* 10.1 (Jan. 2023), p. 1.

[11] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[12] *LocalDpMod — flower.ai*. https://flower.ai/docs/framework/ref-api/flwr.client.mod.LocalDpMod.html.

[13] Wes Mckinney. "pandas: a Foundational Python Library for Data Analysis and Statistics". In: *Python High Performance Science Computer* (Jan. 2011).

[14] H. Brendan McMahan et al. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. 2023. arXiv: 1602.05629 [cs.LG]. URL: https://arxiv.org/abs/1602.05629.

[15] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. *The Hidden Vulnerability of Distributed Learning in Byzantium*. 2018. arXiv: 1802.07927 [stat.ML]. URL: https://arxiv.org/abs/1802.07927.

[16] Vinod Nair and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[17] Joseph P Near et al. *Guidelines for evaluating differential privacy guarantees*. Tech. rep. Gaithersburg, MD, 2025.

[18] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[19] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[20] Sashank Reddi et al. *Adaptive Federated Optimization*. 2021. arXiv: 2003.00295 [cs.LG]. URL: https://arxiv.org/abs/2003.00295.

[21] Holger R. Roth et al. "NVIDIA FLARE: Federated Learning from Simulation to Real-World". In: (2022). DOI: 10.48550/ARXIV.2210.13291. URL: https://arxiv.org/abs/2210.13291.

[22] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[23] Alexander Ziller et al. "PySyft: A Library for Easy Federated Learning". In: *Federated Learning Systems: Towards Next-Generation AI*. Ed. by Muhammad Habib ur Rehman and Mohamed Medhat Gaber. Cham: Springer International Publishing, 2021, pp. 111–139. ISBN: 978-3-030-70604-3. DOI: 10.1007/978-3-030-70604-3_5. URL: https://doi.org/10.1007/978-3-030-70604-3_5.