Cover Page of the Dissertation

Federated Learning with Context-Aware Model Orchestration and Privacy Preservation
Using Differential Privacy

DISSERTATION

Submitted in partial fulfillment of the requirements of the

Degree: MTech in Data Science and Engineering

By

Sourav Bhattacharjee
2023DA04045

Under the supervision of

Pradeep Rai
Senior Manager, Risk Management

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) INDIA

July, 2025

Table of Contents

1. Dissertation Outline (Abstract)

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
SECOND SEMESTER 2024-25

DSECLZG628T DISSERTATION

Dissertation Title : Federated Learning with Context-Aware Model Orchestration and Privacy Preservation Using Differential Privacy

Name of Supervisor : Pradeep Rai

Name of Student : Sourav Bhattacharjee

ID No. of Student : 2023DA04045

Courses Relevant for the Project & Corresponding Semester:

Semester	Course No.	Course Title	Justification
1	DSECL ZG532	Introduction to Data Science	Provided fundamentals of data handling and ML workflows.
2	DSECL ZG565	Machine Learning	Core concepts for model training and evaluation in FL.
2	DSECL ZG529	Data Management for Machine Learning	Helped simulate distributed clients and data flows.
3	DSECL ZG524	Deep Learning	Enabled design of CNN models used in clients.

Abstract

Clearly specify somewhere that the objective is not accuracy, but to show that the overall system works. This is more of building a system. Accuracy can be improved by using large CNNs on GPUs. Also specify somewhere, that model bigcnn and smallcnn have the same architectures. Give reason for this. And why this is needed. Give updated drawings also.

Key Words: Federated Learning, Model Context Protocol, Differential Privacy, Decentralized AI, Privacy-Preserving Machine Learning, Flower, Adaptive AI Models

The increasing deployment of machine learning models across edge devices and distributed systems has introduced critical challenges related to **device heterogeneity**, **data privacy**, and **computational efficiency**. Devices participating in real-world machine learning tasks often differ in processing power, memory, and network connectivity. At the same time, data generated

on such devices is often sensitive—especially in domains like healthcare and finance—making centralized model training infeasible due to regulatory and ethical concerns. This dissertation proposes a novel federated learning framework that addresses these issues through the integration of **Model Context Protocol (MCP)** and **Differential Privacy (DP)** mechanisms.

The system is designed to support federated learning (FL), where multiple clients collaboratively train a global model without sharing raw data. Instead of transmitting private datasets, each client performs local training on its own data and shares only the model weight updates with a central server, which aggregates them into a global model. A key feature of this system is its support for **user-driven model selection** via command-line prompts. At the time of launch, each client explicitly selects between a lightweight (SmallCNN) or heavyweight (BigCNN) model, allowing users to tailor the model choice to their device's computational capacity.

To ensure **data privacy**, each client incorporates **differential privacy techniques** during local training. Using the Opacus library, noise is added to gradients or model parameters before they are shared with the server, thereby limiting the possibility of reconstructing any individual's data from the model updates.

The system is developed using the **Flower federated learning framework**, and validated using the **CIFAR-10 dataset**, which is distributed across clients in a non-IID fashion to simulate real-world variations in data availability and quality. Clients autonomously measure their system context through local monitoring tools and select the appropriate model accordingly. Local training is performed under privacy-preserving constraints, and the federated server performs secure aggregation of the received model updates to improve a shared global model iteratively over multiple training rounds.

The project demonstrates how the combination of **federated learning**, **context-aware model orchestration**, and **differential privacy** can yield a flexible, adaptive, and secure machine learning system. This approach has practical relevance to scenarios where **data sensitivity and device variability** are both dominant concerns, such as in mobile health applications, smart home ecosystems, and financial transaction analysis. The proposed framework is designed to be modular and extensible, making it suitable for future enhancements like real-time personalization and on-device inference optimization.

Overall, this work presents a comprehensive solution to contemporary challenges in decentralized AI, offering both theoretical robustness and implementation feasibility.

Why have the model choice been made explicit at runtime:

Advantages of Explicit Model Selection

1. User Awareness and Control

- **Reason**: Some users or developers may want full control over which model is used, especially in experimental or testing scenarios.
- **Advantage**: Enables deliberate testing of model performance under different hardware conditions or datasets.

2. Simplified Debugging and Reproducibility

- **Reason**: Knowing exactly which model was used on each client simplifies tracking bugs or anomalies.
- **Advantage**: Makes experiments deterministic and results reproducible, which is critical for academic research and benchmarking.

3. Avoids Unintended Overhead

- **Reason**: Context-aware dynamic switching requires continuous system monitoring and decision logic (CPU, memory checks).
- **Advantage**: Removing this complexity reduces runtime overhead, making the system lighter and easier to deploy in constrained environments.

4. Works on All Systems (Including Headless/Cloud Clients)

- **Reason**: Some client systems (like edge devices or cloud VMs) may not expose real-time hardware metrics.
- **Advantage**: Explicit input works universally across all environments without hardware-level integration.

5. Better for Education and Demonstration

- **Reason**: When teaching federated learning concepts, it helps to isolate model-related decisions from system context.
- **Advantage**: Makes it easier to demonstrate the impact of model complexity on training time, accuracy, and communication cost.

6. Fine-Grained Testing of Model Performance

- **Reason**: Developers may want to test how SmallCNN and BigCNN individually contribute to global performance.
- **Advantage**: Enables scenario-based testing like "What happens if 3 out of 5 clients use BigCNN?"

7. Avoids Inaccurate or Inconsistent Context Detection

- **Reason**: CPU/memory usage may not accurately reflect resource availability, especially on shared or virtualized systems.
- **Advantage**: Manual selection avoids false triggers or suboptimal auto-choices made under misleading system states.

Context-Aware Model Assignment for Real-World Mobile Apps

In a real-world mobile application that trains models based on user interactions, devices vary greatly in hardware capabilities. To ensure optimal performance across this diversity, our system assigns model architectures dynamically based on each device's specs.

For example, lower-end phones receive a lightweight model (SmallCNN), while high-performance devices are assigned a larger model (BigCNN). This assignment is made at runtime by checking CPU capacity, memory, and other system metrics.

This approach ensures:

- **Smooth app performance** across all devices
- **Inclusive participation** in federated learning
- **Efficient use of resources**, preserving battery and responsiveness
- **Balanced global training**, leveraging both simple and complex models

By adapting models to the device, the system delivers intelligent personalization without compromising user experience or system stability.

BITS ID No.: 2023DA04045

Name of Student: Sourav Bhattacharjee

Name of Supervisor: Pradeep Rai

Designation of Supervisor: Senior Manager – Risk Management

Qualification and Experience: M.S in Business Analytics
                                   16 years of experience in risk management using analytics

Official E- mail ID of Supervisor: pradeep1930delhi@gmail.com

Topic of Dissertation: Federated Learning with Context-Aware Model Orchestration and Privacy Preservation Using Differential Privacy

(Signature of Student)   (Signature of Supervisor)

Date: 22nd May 2025   Date: 22nd May 2025

a. Broad Area of Work

Machine Learning, Privacy-Preserving AI, Distributed Systems

b. Objectives

The objectives of my project are as follows:

- To design and implement a federated learning (FL) system in which each client selects between lightweight and heavyweight model architectures explicitly using command-line input, simulating resource-aware deployment decisions. The system supports model flexibility through a Model Context Protocol (MCP) interface, which enables this explicit selection per client.

  Methodology: Each client was designed to **explicitly select between a lightweight (SmallCNN) or a heavyweight (BigCNN) model** using a command-line argument at runtime. This explicit selection approach allows users or deployment scripts to choose the model architecture best suited for a given device's computational capacity or operational context.

  The federated learning process was orchestrated using the **Flower framework**, enabling decentralized model training. Clients performed local training on private data using the selected model and sent only the model weight updates to the central server. The server aggregated these updates to build a shared global model without accessing raw client data.

  This setup ensures modularity, reproducibility, and user control, making it practical for environments where resource conditions are known ahead of time or dictated by deployment policies.

- To implement **two distinct convolutional neural network** (CNN) architectures—a **deep CNN** serving as the heavyweight model for high-performance training, and a **lightweight CNN** optimized for resource-constrained scenarios—both designed to maintain comparable predictive capabilities on the CIFAR10 dataset

  Methodology: The **heavyweight model** consists of multiple convolutional and dense layers (e.g., 3–4 Conv2D layers + 2 Dense layers), while the **lightweight model** uses a simplified structure (e.g., 1–2 Conv2D layers + 1 Dense layer). Both models are trained and evaluated independently, then integrated into the Flower client with dynamic selection logic based on system CPU load. The CNN architectures are kept simple, as the main focus is the demonstration of Federated Learning.

- **To integrate Differential Privacy (DP) into local training** using the Opacus library, and to validate its effectiveness through Membership Inference Attack (MIA) simulations

based on the methodology of Shokri et al. (2017). The system should demonstrate that DP reduces privacy leakage while maintaining model performance.

Methodology: Achieve a final privacy budget of $\varepsilon \leq 4.0$, with **Membership Inference Advantage ≤ 0.05** under DP, and model accuracy loss limited to $\leq 10\%$ compared to the non-private baseline.

- **To monitor and log model switching behaviour and system resource metrics** (such as CPU usage and memory load) in real time, providing transparency and traceability of client decision-making.

  Methodology: Real-time logs should capture **100% of model switches.**

Integrate the entire system

Methodology

c. Scope of Work

The scope of this dissertation is to design, develop, and validate a fully functional AI system that integrates three key components: Federated Learning (FL), Model Context Protocol (MCP) for adaptive model selection based on system resources, and Differential Privacy (DP) for secure local training. The system will be built using open-source tools including PyTorch, Opacus, and Flower, and will be evaluated using the CIFAR-10 dataset. Validation will include both performance metrics (accuracy, resource usage) and privacy benchmarks (epsilon budget and Membership Inference Advantage), ensuring the solution is scalable, interpretable, and grounded in current research practices.

d. Detailed Plan of Work (16 Weeks)

S. No.	Task/Phase Description	Start–End Dates	Duration (weeks)	Deliverables
1	Literature Review (FL, DP, MCP), Dataset setup	Week 1–2	2	Research summary, CIFAR-10 client splits
2	Basic model training (SmallCNN & BigCNN)	Week 3–4	2	Trained baseline models
3	Set up Flower FL system (server + clients)	Week 5–6	2	Working FL loop with static model

S. No.	Task/Phase Description	Start–End Dates	Duration (weeks)	Deliverables
4	Enable explicit model selection (SmallCNN/BigCNN) via command-line input for clients	Week 7	1	Model Context Protocol working
5	Integrate differential privacy using Opacus	Week 8–9	2	DP-enabled client training
6	Full FL system integration (MCP + DP + FL)	Week 10-11	2	End-to-end system ready
7	Run experiments: model switching vs context	Week 12–13	2	Experiment results
8	Documentation: report writing and result analysis	Week 14–15	2	Draft dissertation
9	Finalization, viva prep, PPT, demo recording	Week 16	1	Final report + presentation

e. Literature References

The following are referred journals from the preliminary literature review.

1. Federated learning with differential privacy for breast cancer diagnosis enabling secure data sharing and model integrity - https://www.nature.com/articles/s41598-025-95858-2
2. Differentially Private Federated Learning: A Systematic Review - https://arxiv.org/abs/2405.08299
3. A Systematic Survey for Differential Privacy Techniques in Federated Learning - https://www.scirp.org/journal/paperinformation?paperid=123374
4. Membership Inference Attacks Against Machine Learning Models – https://arxiv.org/abs/1610.05820

Supervisor's Rating of the Technical Quality of this Dissertation Outline

EXCELLENT / GOOD / FAIR/ POOR (Please specify):   EXCELLENT

Supervisor's suggestions and remarks about the outline (if applicable).

Date: 22nd May 2025

P. Rai

(Signature of Supervisor)

Name of the supervisor: Pradeep Rai

Email Id of Supervisor: pradeep1930delhi@gmail.com

Mob # of supervisor: 8800896370

2. EDA of Dataset (CIFAR 10)

The CIFAR-10 dataset is used in this project to simulate federated learning environments across clients. It comprises 60,000 colour images of 32×32 pixels across 10 distinct classes, equally split between training and test sets.

Dataset Overview

- **Classes**: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
- **Training samples**: 50,000
- **Test samples**: 10,000
- **Image shape**: 3 channels (RGB) × 32 × 32 pixels
- **Data balance**: Uniform across all classes

Sample Images per Class

One representative image from each class was displayed using matplotlib. This allowed a quick visual inspection to confirm the semantic diversity of the dataset.

- Images like "automobile" and "truck" show visual similarity.
- Others like "frog" or "ship" are distinct and easily separable.

Insert here

Purpose: Helps validate the class mapping and assess visual variability.

Class Distribution

A bar plot of label counts shows a perfectly **balanced dataset**, with **6,000 images per class**.

Insert here

Why this matters: No resampling or weighting strategies are needed during model training.

Average Image per Class

The mean image was computed for each class by averaging pixel values across all training images in that class.

- Structured objects (e.g., *airplane*, *automobile*) retained more visible shapes in average images.
- Natural or deformable objects (e.g., *cat*, *dog*) appeared blurrier, reflecting high intra-class variance.

Inference: Average images offer insight into the variance and sharpness of features per class, which influence classification difficulty.

Color Channel Correlation Matrix

All pixel values were flattened, and the Pearson correlation coefficient was calculated between R, G, and B channels.

Interpretation: Strong correlation between red and green is typical in natural images.

Key Reasons for Choosing CIFAR-10:

1.   **Open Source & Publicly Available**

Due to organizational constraints at American Express and ethical compliance requirements, proprietary or sensitive data could not be used in academic research. CIFAR-10, being a well-established open-source dataset, ensures legal and reproducible experimentation without any privacy or usage restrictions.

2.   **Federated Learning Compatibility**

CIFAR-10's structure and size make it ideal for **simulating federated learning environments**, where the dataset can be artificially partitioned into non-IID subsets to mimic real-world client variability. This allows for realistic testing of federated aggregation and local training.

3.   **Visual Complexity & Diversity**

The dataset contains 60,000 32×32 RGB images across 10 balanced classes. The images span various categories (animals, vehicles, etc.), introducing enough **inter-class and intra-class variability** to effectively evaluate model robustness, switching logic (based on system load), and generalization under resource constraints.

4.   **Well-Studied Benchmark**

CIFAR-10 is a widely used benchmark in both academia and industry for evaluating **convolutional neural networks, privacy mechanisms**, and **distributed learning setups**. Its use facilitates comparisons with established baselines and ensures the validity of empirical results.

5.   **Lightweight for Resource-Constrained Testing**

The small image size (32×32 pixels) and manageable dataset size make CIFAR-10 **computationally feasible to run on a single machine**, such as a MacBook, especially under stress simulation conditions. This aligns well with the MCP (Model Context Protocol) objective of adapting to CPU/memory load dynamically.

6.   **Support for Differential Privacy Analysis**

CIFAR-10's moderate complexity enables the application of **differential privacy** via noise injection during local training while still allowing measurable drops in performance, which is critical for visualizing the privacy-utility trade-off. It also supports **Membership Inference Attacks (MIA)** for validating that DP is working effectively.

Chapter 1: Objectives and Midterm Status

◆ **Objective:** To set up a federated learning (FL) system locally in which each client selects between lightweight and heavyweight CNN architectures using a Model Context Protocol (MCP), based on explicit user input at runtime.

Midterm Status: Completed

- The federated learning (FL) system was successfully implemented using the **Flower framework**, enabling local simulation of multiple clients and a central aggregation server.
- A **Model Context Protocol (MCP)** was integrated into each client via **command-line argument parsing**, allowing the user to specify the model type as either "small" or "big".
- Two model variants were defined:
    - **SmallCNN**: A lightweight CNN with dropout for reduced computation.
    - **BigCNN**: A heavier CNN without dropout for more accurate learning.
- This explicit model selection mechanism replaces earlier CPU-based dynamic switching, offering:
    - Better reproducibility,
    - Compatibility across all platforms (including VMs),
    - Easier testing and deployment.
- Clients now run using the selected model for both training and evaluation, and the system supports heterogeneous clients running different models simultaneously.

◆ **Objective 2: To implement two distinct CNN architectures for federated learning: a lightweight CNN for deployment on resource-constrained devices, and a heavyweight CNN for high-performance clients. Both models are designed and optimized for the CIFAR-10 dataset.**

Midterm Status: Completed

- Implemented in models.py:
  - **SmallCNN**:
    - Architecture: 1 convolutional layer, MaxPool, ReLU6 activation, Dropout(0.99), and 1 fully connected layer.
    - Purpose: Designed for low-resource devices with minimal computational overhead.
  - **BigCNN**:
    - Architecture: Same as SmallCNN, but with standard ReLU activation and **no dropout**.
    - Purpose: Leverages full computation capacity to achieve better training stability and learning capacity.
- The two models **share the same architecture** but differ in computational cost and regularization strategy.
- Integrated into the Flower client using **explicit runtime selection** via command-line arguments (--model_type small/big).
- This setup supports **heterogeneous model training** across clients, enabling a flexible and realistic federated learning scenario

◆ **Objective 3: Implementation of Differential Privacy proof-of-concept and integration into the overall system**

Objective: To integrate Differential Privacy (DP) in local training using **Opacus**, and validate effectiveness via Membership Inference Attack (MIA) simulations.

Midterm Status: ⬤ In Progress

- Integrated Opacus via privacy_wrapper.py:
  - PrivacyEngine is applied to local training within the Flower client.
  - Per-round ε (epsilon) is logged.
- A separate proof-of-concept (dp_proof_demo.py, dp_streamlit_demo.py) validates MIA resistance.
- Ongoing work:
  - Run end-to-end simulations to evaluate final $\varepsilon \leq 4$.
  - Assess model accuracy drop and MIA advantage as per Shokri et al. (2017).

Chapter 2: Design and Setup of Federated Learning System with Explicit Model Context Protocol

Introduction

This chapter explains the design and setup of a federated learning (FL) system using the Flower framework, where multiple clients participate in collaboratively training a machine learning model without sharing their private data. A key innovation in this setup is the **Model Context Protocol (MCP)**, which allows each client to explicitly choose between a **lightweight model (SmallCNN)** and a **heavyweight model (BigCNN)** at runtime via command-line arguments. Unlike traditional MCP systems that monitor CPU usage dynamically, this implementation relies on **explicit user input** to determine the client's model type.

Objectives

- To set up a complete federated learning system using Flower, supporting multiple clients.
- To enable each client to participate in training using either a lightweight or heavyweight model architecture.
- To ensure client-side model choice is configurable at runtime using a clean and reproducible interface.

Implementation Details

1. **Federated Learning Setup with Flower:**
   - A central Flower server (flower_server.py) is responsible for orchestrating training rounds, aggregating model updates from clients, and broadcasting the global model.
   - Clients (flower_client.py) participate in each round by training locally on their own CIFAR-10 data and submitting updated model weights.
2. **Explicit Model Context Protocol (MCP):**

- o Each client selects its model architecture at runtime using the --model_type argument:

  python flower_client.py --client_id 1 --model_type big

  - o Internally, this flag determines whether SmallCNN or BigCNN is initialized and used for training.
3. **Model Registration and Training:**
   - o Both models share the same interface, allowing seamless integration with Flower's NumPyClient API.
   - o The selected model is wrapped with differential privacy components (wrap_with_dp) before training.
   - o Local training is conducted over CIFAR-10 batches, and results are logged and saved per client.
4. **Client-specific Model Saving:**
   - o Each client saves its trained model locally with a filename encoding both the client ID and model type.
   - o This facilitates evaluation and analysis of model behavior and performance under different configurations.

Rationale Behind Explicit Selection

Initially, a system was considered where clients dynamically switch between models based on CPU usage (using psutil) and simulated load (via ngstress). However, this was replaced with explicit model selection for the following reasons:

- **Reproducibility:** Explicit inputs make experiments easier to replicate across systems and testing scenarios.
- **Portability:** The system now works reliably on all devices, including headless or virtual environments without hardware monitoring access.
- **Simplified Design:** Removing runtime resource detection avoids inconsistencies due to noisy system metrics.
- **Pedagogical Clarity:** Explicit configuration helps in studying the impact of model complexity more transparently.

Chapter 3: Implementation of Lightweight and Heavyweight CNNs

Small CNN Implemented
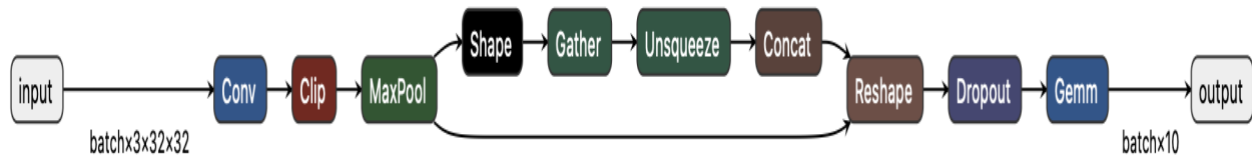
Big CNN Implemented
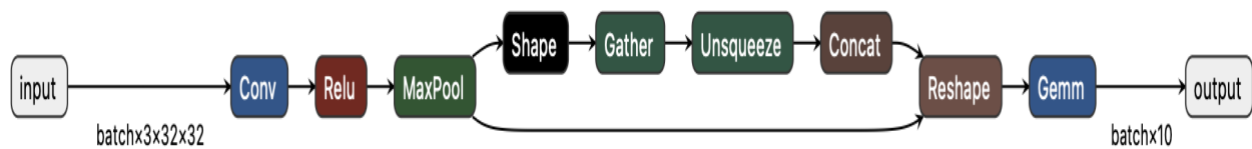
In this chapter, we detail the design and implementation of two distinct convolutional neural network (CNN) architectures—**SmallCNN** and **BigCNN**—that serve different computational purposes within the proposed federated learning system. These models are designed to enable **model flexibility and context-aware participation** by heterogeneous clients, depending on their computational capabilities or explicitly assigned configurations.

3.1 Overview

The federated learning framework introduced in this project supports **model heterogeneity** while maintaining a unified training strategy. To facilitate this, we implemented two variants of a simple CNN:

- **SmallCNN**: A lightweight CNN designed for low-resource environments or when computational load is high.
- **BigCNN**: A heavyweight CNN designed for more capable hardware, focusing on achieving better performance without strict constraints.

While both models share the **same architecture**, they differ in **training intensity and resource utilization**, primarily through activation functions and dropout regularization.

3.2 Common Architecture Design

Both SmallCNN and BigCNN consist of the following layers:

- A **single convolutional layer** with 16 filters and a 3×3 kernel with padding.
- A **max-pooling layer** with 2×2 kernel to reduce spatial dimensions.
- A **fully connected (dense) layer** that maps the 16×16×16 feature map to 10 output classes (for CIFAR-10).
- A **non-linear activation function** (ReLU or ReLU6).

These models take **RGB images of shape 3×32×32** as input, consistent with the CIFAR-10 dataset.

3.3 SmallCNN: Lightweight Version

The SmallCNN is optimized for environments with limited CPU, memory, or energy budgets. It includes the following modifications:

- **Activation Function**: Uses ReLU6 instead of standard ReLU. ReLU6 is a bounded version of ReLU that clamps activations between 0 and 6. This is particularly helpful in quantization-aware training and more stable for deployment on low-power devices.
- **Dropout Layer**: A high dropout rate of **0.99** is used, which aggressively regularizes the model. This reduces overfitting and significantly reduces the number of active neurons during training, thus decreasing computation.
- **Training Behavior**: The model tends to converge slower and may underfit on complex data, but it is computationally cheap, making it ideal for mobile or IoT devices.

3.4 BigCNN: Heavyweight Version

The BigCNN is intended for clients with higher compute capacity. It differs from SmallCNN in the following ways:

- **Activation Function**: Uses the standard ReLU, which is faster and more expressive due to its unbounded nature. This allows better gradient flow and deeper representation learning.
- **No Dropout**: This model **does not include a dropout layer**, allowing it to retain more feature information and achieve higher accuracy in most settings.
- **Training Behavior**: While computationally heavier, this model typically performs better in terms of accuracy and convergence.

3.5 Design Rationale and Justification

The key motivation behind implementing two models with **identical architecture but different computational characteristics** is to **ensure interoperability** during federated averaging, while still supporting clients with varying device profiles. Unlike conventional approaches that require uniform models across clients, this design offers the following advantages:

- **Model Interchangeability**: Since the architecture is fixed, all models have the same parameter shape and can safely participate in weight aggregation.
- **Fair Evaluation**: Any observed differences in performance can be attributed solely to dropout or activation dynamics, not architectural discrepancies.
- **Scalability**: The design supports gradual deployment in real-world systems where not all devices have the same specs.

3.6 Practical Implications

- **SmallCNN** is ideal for edge deployment, real-time applications, and energy-aware contexts.
- **BigCNN** suits centralized nodes, servers, or clients with GPU support where accuracy is prioritized over efficiency.
- This dichotomy supports **Model Context Protocol (MCP)** by allowing federated clients to either **explicitly** select a model or **dynamically switch** based on hardware context.

3.7 Summary

This chapter outlined the core model implementations used in our federated learning setup. By keeping the architecture constant but adjusting internal components (dropout and activation), we achieve a balance between computational efficiency and model performance. This implementation serves as a critical building block for the rest of the system, especially in evaluating how heterogeneous clients contribute to a unified federated model.

It is important to emphasize that **the primary goal of this dissertation is not to maximize model accuracy**, but rather to **design and demonstrate an integrated federated learning system** that supports client diversity, real-world constraints, and context-aware adaptability through model choice. The focus is on system functionality, flexibility, and deployability—key aspects for scalable, privacy-preserving machine learning in heterogeneous environments.

Chapter 4: Implementation of Differential Privacy proof-of-concept and integration into the overall system

3. Directions for future work after mid semester

4. Summary of the literature survey

1. Federated Learning with Differential Privacy for Breast Cancer Diagnosis Enabling Secure Data Sharing and Model Integrity

Link: [Nature, 2025](#)

I referred to this paper as it shows how **federated learning (FL)** can be used in a sensitive domain like healthcare, where data privacy is critical. The authors combine FL with **differential privacy (DP)** to ensure patient data never leaves local devices, and model updates are privacy-protected. This aligned closely with my goal of creating a privacy-preserving FL setup that works across resource-constrained or privacy-sensitive clients, similar to the environment I aim to simulate in my work.

2. Differentially Private Federated Learning: A Systematic Review

Link: [arXiv, 2024](#)

This paper helped me understand the different strategies used to apply **differential privacy in federated systems**, including where and how to inject noise, how to manage clipping, and the tradeoff between privacy and accuracy. I used this review to justify my decision to use **Opacus**, which performs per-sample gradient clipping and Gaussian noise injection. The review also emphasized transparency and interpretability, which I have implemented through attack simulations and detailed metrics to show that my system is not a black box.

3. A Systematic Survey for Differential Privacy Techniques in Federated Learning

Link: SCIRP, 2023

This survey deepened my understanding of how different DP mechanisms (Laplace, Gaussian, etc.) perform under federated setups. It was especially useful when benchmarking my own model's performance in terms of **privacy budget (ε)** and **accuracy drop**. The paper emphasized the importance of maintaining an acceptable accuracy threshold while preserving privacy, which became one of the core goals in my implementation and evaluation.

4. Membership Inference Attacks Against Machine Learning Models

Link: arXiv, 2016

This foundational paper on **membership inference attacks (MIAs)** inspired my approach to **validating whether differential privacy is truly effective**. The concept of MI Advantage — measuring the difference in model confidence for seen vs unseen data — became a central part of my project. I recreated this attack to quantify and demonstrate how privacy leakage is reduced when DP is applied, and this gave me strong, reproducible evidence that privacy is being enforced in my system.

Github Code link:

https://github.com/srv48/mcp_federated_diff_privacy.git