# **Partitioning Techniques in AI Models for Wireless Network Optimization**

**Distributed Machine Learning**

|  |  |  |
| --- | --- | --- |
| **NAME** | **BITS ID** | **CONTRIBUTION** |
| SUBHRANSU MISHRA | 2023AC05489 | 100% |
| DULAL DAS | 2023AC05041 | 100% |
| LAKSHMISRINIVAS PERAKAM | 2023AC05540 | 100% |
| ARCHAN GHOSH | 2023AC05042 | 100% |

# **Part 1: Conceptual Understanding: Definitions in AI Model Design**

# **Vertical Partitioning**

## **Definition**: Vertical partitioning involves splitting the **features (input variables)** across multiple **sub-models** or **computational resources**. Each sub-model receives **only a subset of features** and processes them **independently**.

## **Context in AI**: Each node or sub-model receives a subset of features for the same set of samples. This is common in **cross-silo federated learning**, where institutions (e.g., hospitals, telecom providers) hold different types of data about the same users.

## **Example**: In a wireless network, one node might process signal strength and frequency features, while another handles user mobility and device type.

In **wireless network optimization**, we might:

* Split **network parameters** (e.g., signal strength, network traffic) into one sub-model processed at an **edge node**.
* Process **user attributes** (e.g., user count, device type) in another sub-model at a **central server**.

The outputs from these sub-models are then **combined to predict overall network latency**.

# **Horizontal Partitioning**

* **Definition**: Horizontal partitioning involves splitting the **dataset into subsets** based on **sample characteristics**.  
  Each subset:
* Contains **all features**
* Has only a **portion of the data samples**
* Is distributed across devices or nodes for **parallel training or inference**

## **Context in AI**: Each node trains on a different subset of users or devices, but with the same feature space. This is typical in **cross-device federated learning**.

## **Example**: In a wireless network, one node might train on data from urban users, another on rural users, each with full feature sets like signal strength, latency, and throughput.

In the wireless scenario:

* Partition data **geographically** or **per tower ID** (e.g., TWR001, TWR002).
* Each tower processes its own **latency and traffic data locally**.
* A central model then **aggregates the insights** from individual tower-based models for global latency prediction.

# **Comparison of Vertical vs Horizontal Partitioning**

Here's a comparative table highlighting key aspects relevant to wireless networks:

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Vertical Partitioning** | **Horizontal Partitioning** |
| **Computational Efficiency** | * **Moderate**; communication overhead between sub-models. * Lower complexity per sub-model. * Ideal when input data dimensionality is high, but compute resources are limited per node. | * **High**; parallel computation enables faster training/inference. * Lower overhead if models run independently. * Ideal when dataset is large and geographically distributed (e.g., different towers). |
| **Scalability** | * **Limited** by complexity of feature combination logic. * Scalability depends on inter-node communication efficiency. * Suitable for fewer, high-capacity nodes (e.g., cloud-edge setups). | * **High** scalability; easily expands by adding more computational nodes/towers. * Suitable for distributed and federated learning setups (e.g., multiple towers/users). |
| **Deployment in Wireless Networks** | * **Complex** deployment due to synchronization needs. * Suitable for scenarios where edge devices have limited processing capacity, and feature subsets must be processed centrally. * Sensitive to network bandwidth and latency. | * **Easier** to deploy due to independent local processing. * Ideal for edge-centric deployments with local computation capabilities (e.g., cell towers handling regional data). * Can function effectively under intermittent connectivity. |

## **Examples and Application in Wireless Network Datasets**

### **Vertical Partitioning Example (Feature-based)**

****Scenario:**** Latency prediction in **rural IoT setups**.

****Implementation:****

* + ****Edge Node (Local):****
    - Processes real-time ****network traffic**** and ****signal strength**** measurements  
      → Features: Signal Strength (dBm), Network Traffic (MB)
  + ****Central Server:****
    - Processes ****user demographic information**** and ****historical patterns****  
      → Features: User Count, Device Type, Location Type

****Advantage:****

* + Edge devices handle **real-time, low-dimensional** processing.
  + Centralized models handle **complex, aggregated, or historical** data.
  + Enables **efficient distribution of computational load**.
* **Horizontal Partitioning Example (Sample-based/Region-based)**

****Scenario:**** Distributed latency prediction models across **multiple wireless towers**.

****Implementation:****

* + Each **wireless tower** (e.g., TWR001, TWR002, TWR003) independently:
    - Trains a model on **locally collected data** with the **full feature set**  
      → Signal Strength, Network Traffic, User Count, Device Type
  + Models are periodically **aggregated using federated learning** techniques.

****Advantage:****

* Enables **parallel training** and **local adaptation**.
* Minimizes **central communication overhead**.
* Well-suited for **scalable and privacy-aware deployments**.

## **Summary Table with Wireless Network Dataset Reference**

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition Type** | ****Example Application (Wireless Networks)**** | ****Data Handling**** | ****Computational Model Setup**** |
| **Vertical** | Feature-based splitting (Edge-Cloud) | Different feature subsets at different locations | Multiple smaller models, combined post-processing |
| **Horizontal** | Tower-based or regional data subsets | Same features across different locations/samples | Independent local models aggregated centrally |

## **Conclusion & Recommendation for Wireless Networks**

* **Use Vertical Partitioning** when:
  + Edge devices have **limited computational power**
  + Central infrastructure can handle **complex, non-real-time computations**
  + There's a clear split in feature roles (e.g., real-time vs. historical)
* **Use Horizontal Partitioning** when:
  + The environment is **highly distributed** (e.g., many towers or IoT nodes)
  + **Local data processing** is beneficial
  + **Scalability and robustness** are important
  + Minimizing central server dependency improves **resilience and autonomy**

These partitioning strategies—when matched to the system’s architecture and data characteristics—can significantly improve model **performance, latency prediction,** and **scalability** in wireless communication systems.

## **Appendix**

## **Examples Using Wireless Network Datasets**

## **Dataset References**

## **SNDLib & Abilene Network**: Provide traffic matrices and topologies useful for partitioning experiments.

## **WACA (Wi-Fi All-Channel Analyzer)**: Captures RSSI across all Wi-Fi channels, ideal for vertical partitioning where each channel is a feature.

## **Google Stadia Traffic Dataset**: Useful for horizontal partitioning based on user behavior or resolution preferences.

## **Real-world RF Dataset (IEEE 802.11ax, LTE, 5G-NR)**: Contains signal traces across technologies, enabling both vertical (modality-based) and horizontal (user-based) partitioning.

## **Use Case Examples**

## **Vertical**: A telecom provider splits features between signal quality metrics and user mobility patterns, training sub-models separately and merging predictions.

## **Horizontal**: A federated learning setup where base stations train models on local user data (e.g., throughput, latency), then aggregate updates centrally.

## **Key Papers Explored**

## **1.** [**Cross-Silo Federated Learning for Multi-Tier Networks**](https://arxiv.org/html/2108.08930v4)

## **Authors**: Anirban Das, Timothy Castiglia, Shiqiang Wang, Stacy Patterson

## **Focus**: Proposes a *Tiered Decentralized Coordinate Descent (TDCD)* algorithm for federated learning across silos with both vertical and horizontal partitioning.

## **Relevance**: Directly addresses wireless network scenarios with distributed data across clients and hubs.

## **2.** [**Vertical and Horizontal Partitioning in Data Stream Regression Ensembles**](https://jpbarddal.github.io/assets/pdf/vertical_horizontal_regression.pdf)

## **Author**: Jean Paul Barddal

## **Focus**: Investigates how combining vertical and horizontal partitioning improves regression accuracy in data stream mining.

## **Relevance**: Useful for understanding ensemble learning in dynamic wireless environments.

## **3.** [**Overview of Horizontal and Vertical Partitioning**](https://www.researchgate.net/profile/Mansi-Bosamia/publication/325570099_Overview_of_Horizontal_Partitioning_and_Vertical_Partitioning/links/5b167bf7a6fdcc31bbf5a2e6/Overview-of-Horizontal-Partitioning-and-Vertical-Partitioning.pdf)

## **Authors**: Meeta J. Pajwani, Mansi P. Bosamia

## **Focus**: Offers a foundational overview of partitioning strategies with implications for query optimization and distributed systems.

## **Relevance**: Good for conceptual grounding before diving into AI-specific applications.

## **4.** [**GeeksforGeeks Comparison Article**](https://www.geeksforgeeks.org/system-design/vertical-partitioning-vs-horizontal-partitioning/)

## **Title**: Vertical Partitioning vs Horizontal Partitioning

## **Focus**: A practical comparison of the two methods in system design.

## **Relevance**: While not a formal paper, it’s a helpful primer for understanding trade-offs.