Norges miljø- og biovitenskapelige universitet

Master's Thesis 2024 30 ECTS
Faculty Of Science and Technology

Anomaly Detection in Mobile Broadband Network Using Federated Learning

Okubadejo Olutomi Samuel
Master Of Science In Data Science

Acknowledgements

First and foremost, I extend my deepest gratitude to God, whose guidance and wisdom have been my constant support throughout this journey.

I am profoundly grateful to my main supervisor, Azza Hassan Mohamed Ahmed from Simula Met, whose expertise and insightful guidance have been pivotal in shaping this thesis. Her dedication and support have been invaluable, and I am deeply appreciative of her mentorship.

I would also like to express my sincere thanks to Jonas Kusch at NMBU for his invaluable contributions and steady encouragement throughout my research process. His advice and expertise have greatly enhanced my work.

My heartfelt appreciation goes to my family, whose endless support and love have been my stronghold. Your belief in me has been a source of motivation and strength.

I am also thankful to my friends for their understanding and companionship. Your encouragement and presence have been a great comfort to me throughout this academic endeavor.

Furthermore, I extend my thanks to the entire team at Simula Met for their collaboration and the supportive environment they provided, which enriched my research experience significantly.

Thank you all for your significant roles in my academic journey and personal growth.

Abstract

The advancement of mobile broadband networks has introduced new challenges in data privacy and the efficiency of anomaly detection. Traditional centralized data processing methods are becoming less effective, calling for innovative approaches like federated learning. This master's thesis investigates the application of federated learning to enhance anomaly detection within mobile networks by utilizing real-world data from multiple locations. The objective is to evaluate its ability to train locally on devices, thereby preserving user privacy and leveraging the data's geographical diversity.

The research methodology involved training machine learning models directly on network nodes without centralizing the data. Over 20 rounds of training, the models demonstrated substantial improvements in detecting network anomalies. Notably, the best-performing round achieved an accuracy of 86.91%, precision of 81.18%, and a consistently high recall of over 99%, resulting in an F1 score of 89.17%. These results confirm the effectiveness of federated learning in maintaining high detection rates while enhancing data privacy.

The findings highlight federated learning's potential to transform network management practices, facilitating secure anomaly detection across different geographical regions.

The thesis concludes with a discussion on the scalability of federated learning models and their potential integration with existing technologies to further improve network functionality and security.

List of Abbreviations

AE Autoencoder

ANN Artificial Neural Network

ARIMA Autoregressive Integrated Moving Average model

Bi-LSTM Bidirectional Long Short-Term Memory

CNN Convolutional Neural Network

DDOS Distributed Denial-of-service

DNS Domain Name System

FedAvg Federated Averaging

FL Federated Learning

FNN Feedforward Neural Network

GRU Gated Recurrent Unit

GTP GPRS Tunnelling Protocol

IOT Internet of Things

KPI Key Performance Indicator

LDAP Lightweight Directory Access Protocol

LSTM Long Short Term Memory

MAE Mean Absolute Error

MSE Mean Squared Error

MTS Multivariate Time Series

NNE Nornet Edge

relu Rectified Linear Unit

RNN Recurrent Neural Networks

RSRP Reference Signal Received Power

RSRQ Reference Signal Received Quality

RSSI Received Signal Strength Indicator

SARIMA Seasonal Autoregressive Integrated Moving Average model

SGD Stochastic Gradient Descent

SNMP Simple Network Management Protocol

SVM Support Vector Machine

tanh Hyperbolic Tangent Function

VAE Variational Autoencoder

WSN wireless sensor networks

XGBoost eXtreme Gradient Boosting

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The advancement of mobile networks, from the rudimentary voice services of 1G to the comprehensive connectivity of 5G, has revolutionized communication and significantly increased the complexity and volume of data transmitted over these networks (Figure 1.1). Looking ahead, the ongoing research into 6G and next-generation networks promises even greater enhancements, potentially introducing capabilities that will further transform connectivity standards, facilitate even faster data transmission, and enable more reliable network coverage. These developments will not only extend the frontiers of digital communication but are also likely to introduce new challenges and complexities in network management and data security (Attaran, 2023; Zontou, 2023).

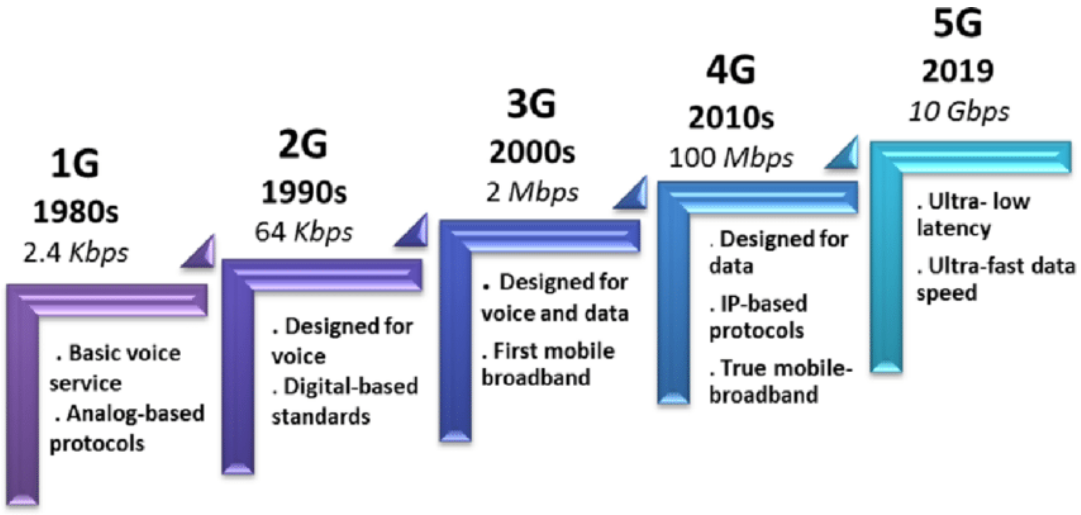


Figure 1.1: The Evolution of mobile networks (Attaran and Attaran, 2020)

The surge in connected devices has increased the complexity and volume of data in contemporary networks. While smartphones remain a major contributor to network traffic, Internet of Things (IoT) devices are rapidly expanding their footprint, generating substantial amounts of data from a lot of sensors and applications (Attaran, 2023; Zontou, 2023). The increase in data volume and complexity has rendered network management a challenge, often exceeding the limits of human

accuracy and efficiency.

To address these challenges, artificial intelligence, and machine learning have become key in network management and security. These technologies are especially useful in mobile networks for spotting trends and detecting unusual activity in large volumes of data.(Alghanmi et al., 2022; Rajendra et al., 2023; Sarker et al., 2021; Taleb et al., 2020).

Advancements in network softwarization, virtualization, and automation have paved the way for leveraging artificial intelligence in mobile networks. These technologies enable more scalable and adaptable networks, optimizing resource utilization. artificial intelligence and machine learning are crucial in refining these advancements and managing the complexities of modern network environments. They enhance network robustness and security against evolving cyber threats, creating a solid infrastructure supporting high data demands and rapid service deployment typical of 5G and future networks (Condoluci and Mahmoodi, 2018; De Turck et al., 2017).

Anomaly detection, a crucial network security component, involves identifying data patterns that deviate from expected behavior. Given the complexity and volume of data in modern networks, particularly with the introduction of 5G, conventional anomaly detection techniques are often insufficient due to privacy concerns, bandwidth limitations, and potential security vulnerabilities (Thudumu et al., 2020; Xu et al., 2022; Yang and Zhang, 2023).

To address these challenges, federated learning offers a solution by facilitating the training of machine learning models on decentralized data. This approach not only addresses the limitations of traditional methods but also significantly enhances privacy and security (Kremenova and Gajdos, 2019; W. Y. B. Lim et al., 2020).

This research applies federated learning to anomaly detection in mobile network real-world data, showcasing its ability to handle complex data and improve privacy. Demonstrating federated learning's capacity to address network anomalies across various locations, the study reveals its adaptability and effectiveness. Such insights show the value of machine learning in enhancing mobile network security and management in real-world applications, thereby contributing to the existing research in the domain.

1.1 Research Question

How can federated learning be effectively applied to real-world mobile network data for anomaly detection, and what insights can be derived from its application across various geographic locations regarding the distribution and frequency of network anomalies?

1.2 Aims and Objectives

The aims and objectives of this research are

1. To explore how federated learning can overcome challenges related to the efficient transfer and processing of high-dimensional data across geographically distributed mobile network nodes.
2. To Investigate the effectiveness of federated learning and other distributed learning techniques in preserving the security of mobile network data.
3. To leverage real-world mobile network data to implement and evaluate federated learning-based anomaly detection, investigating regional disparities in anomaly prevalence.

1.3 Structure of Thesis

Introduction: Introduces the research, highlighting the context, problem statement, and objectives.

Literature Review: Reviews existing research on anomaly detection in mobile networks, highlighting the gap in real-world data applications of federated learning.

Background: Explores the theoretical foundations of machine learning and federated learning, establishing their relevance to anomaly detection in mobile networks.

Methods: Describes the research methodology, including data collection from various locations and techniques employed for federated learning based anomaly detection.

Results: Presents research findings, evaluating the effectiveness of federated learning in detecting anomalies within real-world mobile network data.

Discussion: Discusses the results, comparing federated learning with centralized learning and exploring the implications of regional disparities in anomaly detection.

Conclusion: Summarizes the key findings, contributions, limitations, and future research directions, emphasizing the significance of applying federated learning to real-world data for anomaly detection in mobile networks.

Chapter 2

Background

Machine learning leverages statistics to make decisions in a variety of fields. It is capable of handling complex scenarios and has a quicker response time to abnormal behaviors than humans. It offers flexibility with various algorithms and model choices (S. Wang et al., 2021).

2.1 Categories of machine learning

Machine learning is categorized into three primary types: supervised, unsupervised, and reinforcement learning. Each type provides distinct methods for data analysis and prediction, suitable for different tasks such as anomaly detection, predictive modeling, and pattern recognition. Choosing the correct approach depends on the learning mechanism and data at hand.

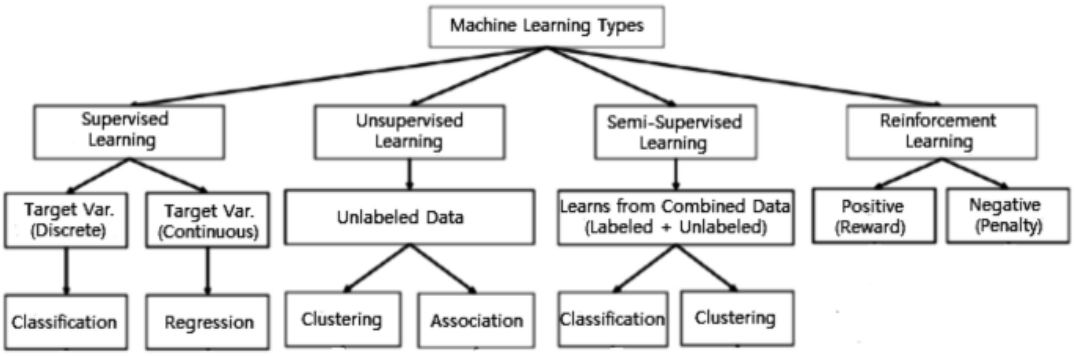


Figure 2.1: Types of Machine learning (Sarker, 2021)

Supervised Learning: It involves training algorithms with labeled datasets. The model learns to map inputs to outputs, preparing it for accurate predictions on new data. It is primarily applied in classification, for sorting data into classes, and regression, for forecasting continuous values.

Techniques include linear Regression for regression, logistic Regression, support vector machines, and neural networks for classification (Cunningham et al., 2008).

Unsupervised Learning: It works with unlabeled data to uncover inherent patterns. Algorithms autonomously discern data relationships, serving well for clustering and dimensionality reduction. K-Means clusters similar data, while Principal Component Analysis streamlines the dataset retaining crucial details (Sutskever et al., 2015).

Semi-Supervised Learning: Semi-supervised learning leverages a small amount of labeled data and a large amount of unlabeled data to improve learning accuracy (Van Engelen and Hoos, 2020).

Reinforcement Learning Reinforcement learning involves an agent learning decision-making through environmental interactions to reach a goal. Rewards and penalties guide the agent, as it associates actions with outcomes to maximize rewards over time. It diverges from supervised and unsupervised learning, prioritizing the exploration-exploitation balance rather than relying on labeled data or pattern detection (Sutton and Barto, 2018).

Deep learning constitutes a subfield within machine learning, contributing to the progression of artificial intelligence. It excels in identifying intricate correlations through the implementation of layered architectures. Employing both supervised and unsupervised learning methods, deep learning forms hierarchical representations of features. These are derived from layers that progressively increase in abstraction (Pang et al., 2021).

Specifically, in the context of time series data, such as that found in mobile networks, deep learning leverages recurrent neural networks. RNNs are adept at incorporating information from preceding time steps during training, which is instrumental in the process of anomaly detection. The capability to extract both temporal and spatial dependencies within the data is a crucial part of this approach.

This chapter outlines the theoretical framework underpinning the application of different deep learning methods for anomaly detection in mobile data. Focused on supervised learning, the chapter details deep learning models prevalent in current research, such as RNN, CNN, and Autoencoders, as reviewed in the literature.

2.2 Artificial Neural Networks

An Artificial Neural Network is a computational framework consisting of interconnected neurons. Typically, an ANN has three layers: an input layer, one or more intermediate layers, and an output layer. Not only do ANNs learn swiftly from data, but they also adeptly handle complex, nonlinear tasks (Sze et al., 2017).

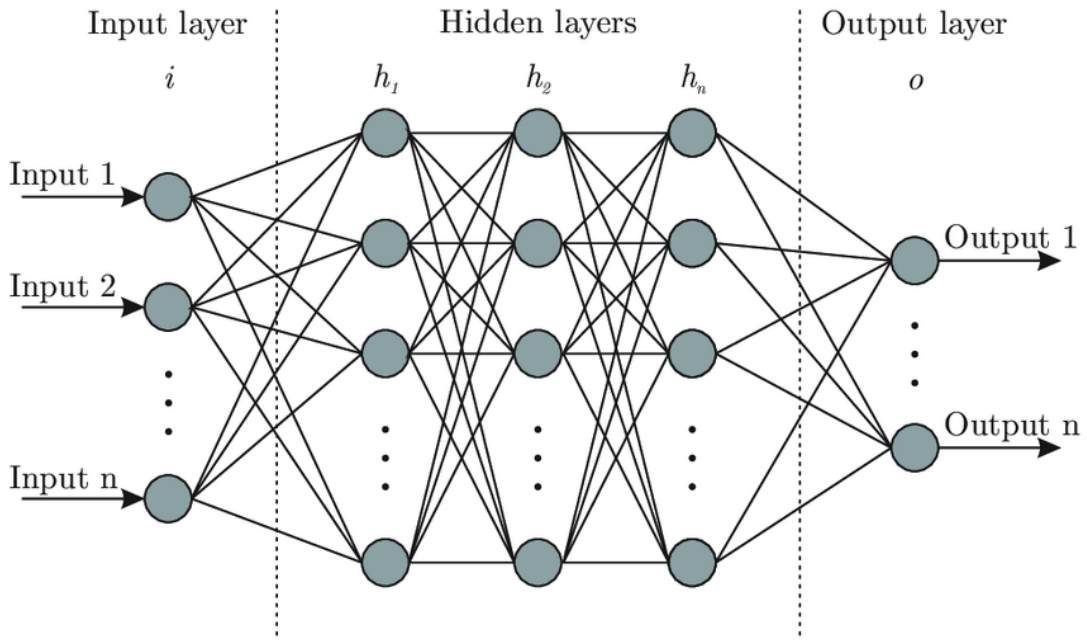


Figure 2.2: Multi Input, hidden and output layers neural network (Bre et al., 2017)

2.2.1 Key Components of ANN:

Here are the key components of ANN (Patel et al., 2022);

Input Layers

The input layer functions as the primary gateway for raw data entering the neural network. It comprises neurons representing individual features of the dataset. If we consider X as the input vector, the operation in this layer can be simply represented as:

$$\text{Input Layer Output} = \mathbf{X} \tag{2.1}$$

This equation signifies that the output from the input layer is exactly the input vector X. It implies that no transformation or computation is being applied at this stage.

Hidden Layer

Hidden layers in a neural network perform essential computations, processing inputs using weighting, biasing, and activation functions to prepare data for the output layer. The network's ability to capture complex representations is determined by the complexity of its architecture, defined by the number and size of hidden layers.

In hidden layers, each neuron applies a weighted sum on the inputs, adds a bias, and then passes the result through an activation function. If X is the input vector, W represents the weights, b is the bias, and f is the activation function, the operation for a single neuron can be represented as:

$$\mathbf{H} = f(\mathbf{W} \cdot \mathbf{X} + \mathbf{b}) \tag{2.2}$$

For multiple neurons from $i to $j in vector form:

$$a = f\left(\sum_{i=1}^{j}(W_i \cdot X_i) + b\right) \tag{2.3}$$

where $f is applied element-wise.

Output Layer

The output layer generates the neural network's result, tailoring it to the task. For classification, it signifies the classes, while for regression, it yields predicted values. The layer's configuration adapts to the application, with activation functions like softmax for multi-class problems or sigmoid for binary classification.

The output layer combines the outputs of the last hidden layer, again applying weights and biases, and often includes a final activation function that is appropriate for the task (e.g., softmax for classification). If H is the output from the last hidden layer, $\mathbf{W_o}$ are the weights, b_o is the bias for the output layer, and f_o is the output activation function, the operation for the output layer is:

$$Y = f_o(\mathbf{W_o} \cdot H + b_o) \tag{2.4}$$

For a classification task with C classes and softmax as the activation function, the formula for the j-th output neuron is:

$$Y_j = \frac{e^{(W_{oj} \cdot H + b_{oj})}}{\sum_{k=1}^{C} e^{(W_{ok} \cdot H + b_{ok})}} \tag{2.5}$$

This softmax function ensures that the output values sum to 1 and can be interpreted as probabilities for each class.

Feedforward Network

Feedforward Neural Networks represent a fundamental neural network model characterized by unidirectional data flow from input to output layers. These networks are structured with an input layer, multiple hidden computational layers, and an output layer specific to tasks like regression or classification. The learning in FNNs relies on weight adjustments through backpropagation, utilizing methods like gradient descent to minimize errors. FNNs are well-regarded for pattern recognition, although they may face challenges with non-linearities and overfitting. Continuous developments in machine learning continue to enhance their application and efficacy in various domains (Michelucci, 2022a).

Loss Function

In training Artificial Neural Networks, loss functions assess the difference between network predictions and actual targets. They're critical for guiding the optimization process, with the goal being loss minimization. For regression, Mean Squared Error (MSE) and Mean Absolute Error (MAE) are standard. MSE emphasizes larger errors

n represents the total number of observations, y_i denotes the actual values from the dataset, and $\hat{y}_i$ indicates the predicted values generated by the regression model for each observation.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.6}$$

while MAE considers all errors equally.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.7}$$

In classification, Cross-entropy loss is preferred for models outputting probabilities, as it penalizes predictions differing from actual class labels (Terven et al., 2023).

Selecting and optimizing the right loss function ensures effective learning and generalization in ANNs.

Activation Functions

Activation functions introduce non-linearity in ANNs, crucial for learning complex patterns. Without them, deep networks simply act linearly (Lederer, 2021).

- Sigmoid: Useful for probabilities but less popular now due to issues like vanishing gradients.
- Tanh: Zero-centered, better for hidden layers as it outputs values from -1 to 1.
- ReLU: Outputs the input if positive, else zero, preferred for its speed and efficiency.
- Softmax: Ideal for multi-class classification in the output layer, converting outputs to probabilities.

The choice of activation function influences ANN training speed, convergence, and generalization ability.

Gradient Descent Algorithm

Gradient Descent is a pivotal algorithm for minimizing the loss function, aiming to find the function's local minimum (Ketkar and Ketkar, 2017). A concise overview of the algorithm is given as;

1. **Initial Parameter Selection:** Initiate each weight and bias within the neural network (NN) with random values. This random initialization is crucial for breaking symmetry, ensuring that hidden layer units learn distinct functions of the input. Without this, identical initial parameters would lead to each unit learning the same input function, negating the network's complexity.

2. **Iterative Parameter Update:** Continue to iteratively adjust the parameters W and b according to the following update rules until reaching a hopefully optimal minimum:

$$W_{l_{i,j}} = W_{l_{i,j}} - \alpha \frac{\partial L(W, b)}{\partial W_{l_{i,j}}}$$

$$b_{l_i} = b_{l_i} - \alpha \frac{\partial L(W, b)}{\partial b_{l_i}}$$

   Here, α represents the learning rate, $W_{l_{i,j}}$ and b_{l_i} denote the weights and biases, respectively, for a given layer l in the neural network, where i represents the index of the neuron in the current layer, and j represents the index of the neuron in the previous layer from which the connection originates. The derivative of the overall loss function with respect to each parameter is calculated as the average of gradients from each training example.

3. **Learning Rate Consideration:** The learning rate α dictates the step size during the descent. A well-chosen α is critical; too small a rate results in slow convergence, while too large a rate might lead to overshooting the minimum or even divergence from the solution.

This iterative process of adjusting neural network parameters is designed to efficiently navigate the complex loss landscape, steering towards a minimum that optimizes the network's performance.

Backpropagation

Backpropagation is a method used for training artificial neural networks, particularly feedforward neural networks (Aggarwal, 2023). Below are the steps and associated formulas:

1. Forward Pass: Compute the output of the neural network for a given input by propagating the input through the network.

   - For each layer l from 1 to L (where L is the last layer):

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}, \quad a^{[l]} = f^{[l]}(z^{[l]})$$

     where $z^{[l]}$ is the pre-activation value, $a^{[l]}$ is the post-activation value (output of layer l), $W^{[l]}$ and $b^{[l]}$ are the weights and biases for layer l, and $f^{[l]}$ is the activation function for layer l.

2. Compute the Loss: Evaluate the loss function to measure the error between the network's output and the actual target values.

- For cross-entropy loss in classification tasks:

$$J(W,b) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{c=1}^{C} y_c^{(i)} \log(a_c^{[L](i)})$$

where J is the cost function, m is the number of training examples, C is the number of classes, $y_c^{(i)}$ is the actual value (target label) for the c-th class of the i-th training example, and $a_c^{[L](i)}$ is the predicted probability of the i-th training example belonging to the c-th class, outputted by the last layer.

3. Backward Pass: Compute the gradient of the loss function concerning each parameter in the network.

   - The gradient for the last layer (L) and any layer l from $L-1$ to 1:

$$\frac{\partial J}{\partial z^{[L]}} = a^{[L]} - y, \quad \frac{\partial J}{\partial W^{[l]}} = \frac{1}{m}\frac{\partial J}{\partial z^{[l]}} a^{[l-1]T}, \quad \frac{\partial J}{\partial b^{[l]}} = \frac{1}{m}\sum \frac{\partial J}{\partial z^{[l](i)}}$$

4. Update the Parameters: Update the weights and biases using gradient descent or its variants.

$$W^{[l]} = W^{[l]} - \alpha\frac{\partial J}{\partial W^{[l]}}, \quad b^{[l]} = b^{[l]} - \alpha\frac{\partial J}{\partial b^{[l]}}$$

   where α is the learning rate.

This method iteratively reduces the error in the network, optimizing its parameters for better accuracy.

2.3 Recurrent Neural Networks

Recurrent Neural Networks are specialized neural networks for pattern recognition within sequences like text or time series data. They have a memory feature that processes sequences, not just single data points. This makes RNNs ideal for applications where context and time are essential. This is the case in our research of high dimensional network data, with a lot of temporal dependencies (Das et al., 2023).

Key Characteristics of RNNs:

- Memory: RNNs maintain information from previous inputs in a sequence through their internal state, which acts as a form of memory. This capacity enables RNNs to display temporal dynamic behavior. Making them valuable for tasks involving sequential data, where context over time is crucial.
- Shared Parameters: weights are utilized consistently across all time steps, reducing the total parameters to be learned and enhancing learning efficiency.

- Variable-Length Sequences: RNNs can handle inputs of varying lengths. This trait is essential for processing sequences like sentences or time series data that don't have a fixed size.

An RNN processes sequences by iterating through the sequence elements. It also maintains a state containing information relevant to what it has seen (Schmidt, 2019).

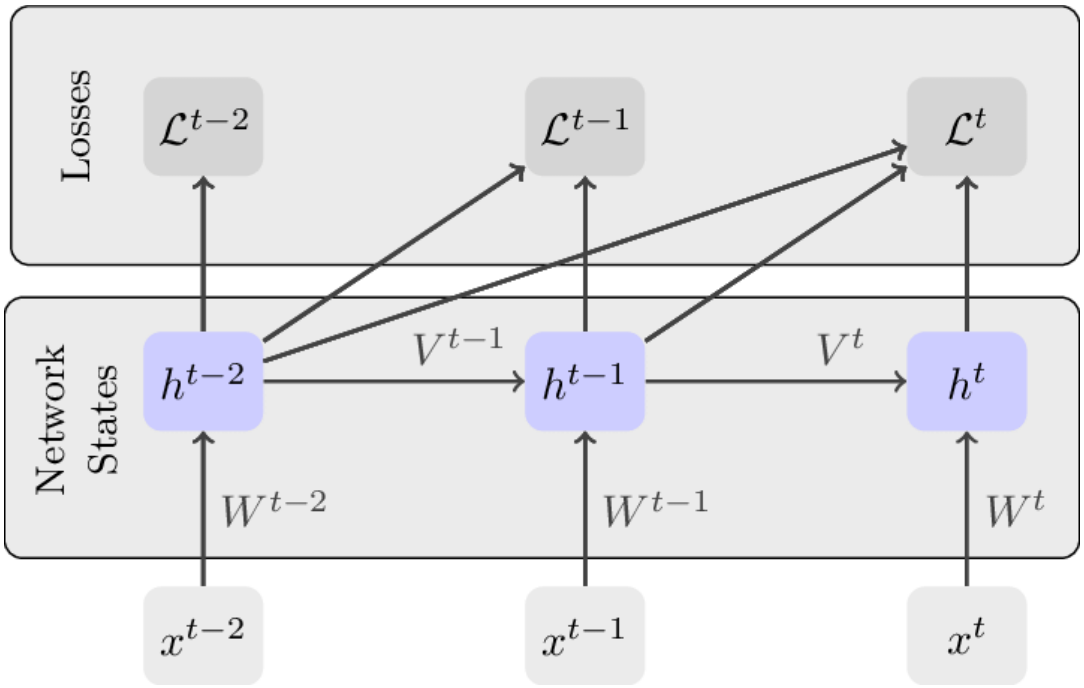


Figure 2.3: Computational graph of RNN (Zenke and Neftci, 2020)

Given the structure of an RNN, the following formulas represent the computational steps across time steps (Sherstinsky, 2020):

1. **Hidden State Update:** At each time step t, the hidden state h_t is updated using the formula:

   $$h_t = f(W_t \cdot h_{t-1} + V_t \cdot x_t + b_h)$$

   W_t represents the recurrent weights applied to the previous hidden state, V_t represents the input weights applied to the current input x_t, b_h is the bias term, and f is the non-linear activation function.

2. **Loss Computation:** The loss L_t at each time step t can be computed as:

   $$L_t = g(h_t, y_t)$$

   g is the loss function measuring the difference between the predicted output and the target y_t, and h_t is the current hidden state.

3. **Backward Pass:** During the backward pass, the gradients of the loss concerning the weights and biases are calculated for each time step:

$$\frac{\partial L_t}{\partial W_t}, \quad \frac{\partial L_t}{\partial V_t}, \quad \frac{\partial L_t}{\partial b_h}$$

These gradients are utilized to update the parameters using gradient descent.

The cyclic dependencies in these computations enable the RNN to process sequences of data by maintaining a state across time steps.

Challenges with RNNs:

Vanishing and Exploding Gradients: RNNs are prone to these issues due to their deep connections across time. This can make training deep RNNs challenging and affect their ability to learn long-term dependencies (Hochreiter, 1998).

Computational Intensity: The sequential nature of RNNs means that inputs need to be processed one after another, making parallelization difficult and often leading to longer training times (Korsky and Berwick, 2019).

2.3.1 Variants and Solutions

Several RNN variants have been developed to address these challenges, including Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs). These architectures solve the vanishing gradient problem and capture long-term dependencies effectively.

Long Short-Term Memory

Long Short-Term Memory Networks, also known as LSTMs, are a specialized form of Recurrent Neural Networks (RNNs) that have the ability to learn and remember long-term dependencies in data. Introduced by Hochreiter and Schmidhuber, 1997, LSTMs were designed to overcome the vanishing gradient problem present in traditional RNNs.

LSTMs are neural networks that use gates to regulate the flow of information. The gates decide which data to keep, allowing LSTMs to pass relevant information down the sequence and make accurate predictions (Van Houdt et al., 2020).

An LSTM unit is composed of a cell (which carries information across many time steps) and three types of gates:

Forget Gate (f): Decides what information is discarded from the cell state.

Input Gate (i): Updates the cell state with new information.

Output Gate (o): Determines what the next hidden state should be.

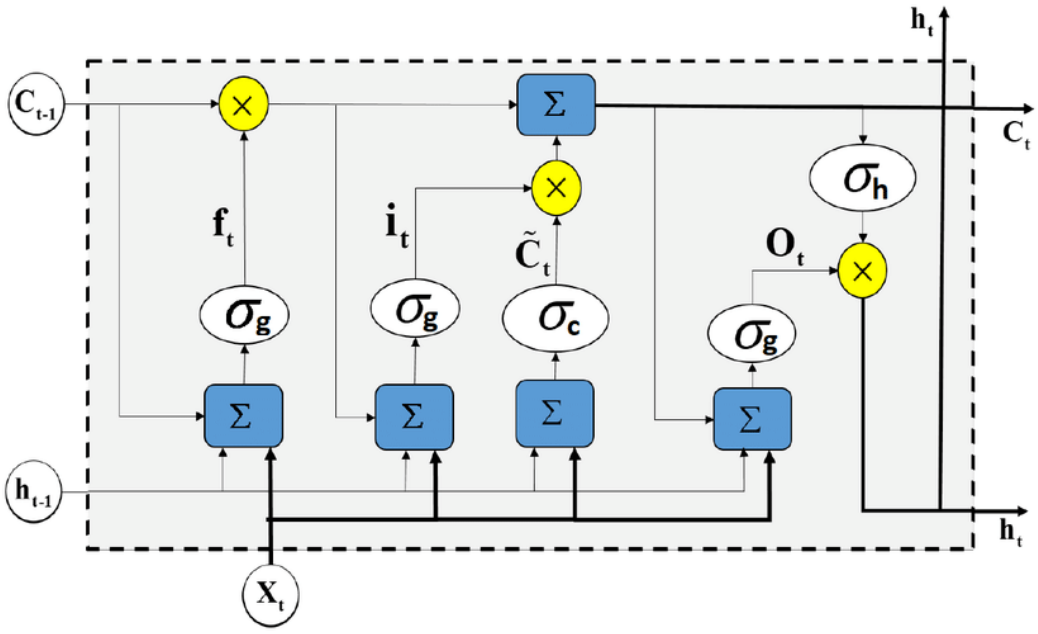


Figure 2.4: Typical LSTM network structure (ElMoaqet et al., 2020)

The LSTM's Internal Mechanisms:

During the forward pass, at each time step, the LSTM updates its internal state through a series of steps (Staudemeyer and Morris, 2019):

Forget Gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

It decides what information to discard from the cell state, using a sigmoid function σ that outputs numbers between 0 and 1.

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

It determines which new information is stored in the cell state, combining a sigmoid layer and a tanh layer to create a vector of new candidate values.

Update Cell State:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

It updates the old cell state C_{t-1} into the new cell state C_t by element-wise multiplying the old state by f_t, discarding the information forgotten, and element-wise adding $i_t * \tilde{C}_t$ for the new candidate values.

Output Gate and Producing the Hidden State:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

The output gate decides what the next hidden state should be. The hidden state contains information about the previous inputs, considering the current input and the memory of the cell.

The output gate o_t is computed by applying the sigmoid function σ to the result of element-wise addition of the weighted sum $W_o \cdot [h_{t-1}, x_t]$ and the bias b_o.

The hidden state h_t is then produced by element-wise multiplying the output gate o_t with the hyperbolic tangent of the cell state C_t.

Gated Recurrent Units

Gated Recurrent Units (GRUs) are a variant of Recurrent Neural Networks (RNNs) introduced by Cho et al., 2014. GRUs are designed to adaptively capture dependencies of different time scales in sequential data, similar to Long Short-Term Memory (LSTM) networks but without using separate memory cells. They are particularly effective in tasks where the sequence length varies and are known for being more computationally efficient than LSTMs (Chung et al., 2014).

Key Characteristics of GRUs:

Simplified Architecture:

GRUs simplify LSTM by combining forget and input gates into a single update gate and merging the cell state and hidden state, resulting in a streamlined model (Shiri et al., 2023).

Update and Reset Gates:

Update Gate: Decides the extent to which the new information will overwrite the existing content in the memory cell

Reset Gate: Decides how much is discarded of the past information.

- The **reset gate** r_t:

  $$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

  This gate determines how much of the past information to forget.

- The **update gate** u_t:

  $$u_t = \sigma(W_u \cdot [h_{t-1}, x_t] + b_u)$$

  This gate decides the amount of the past information to be carried forward.

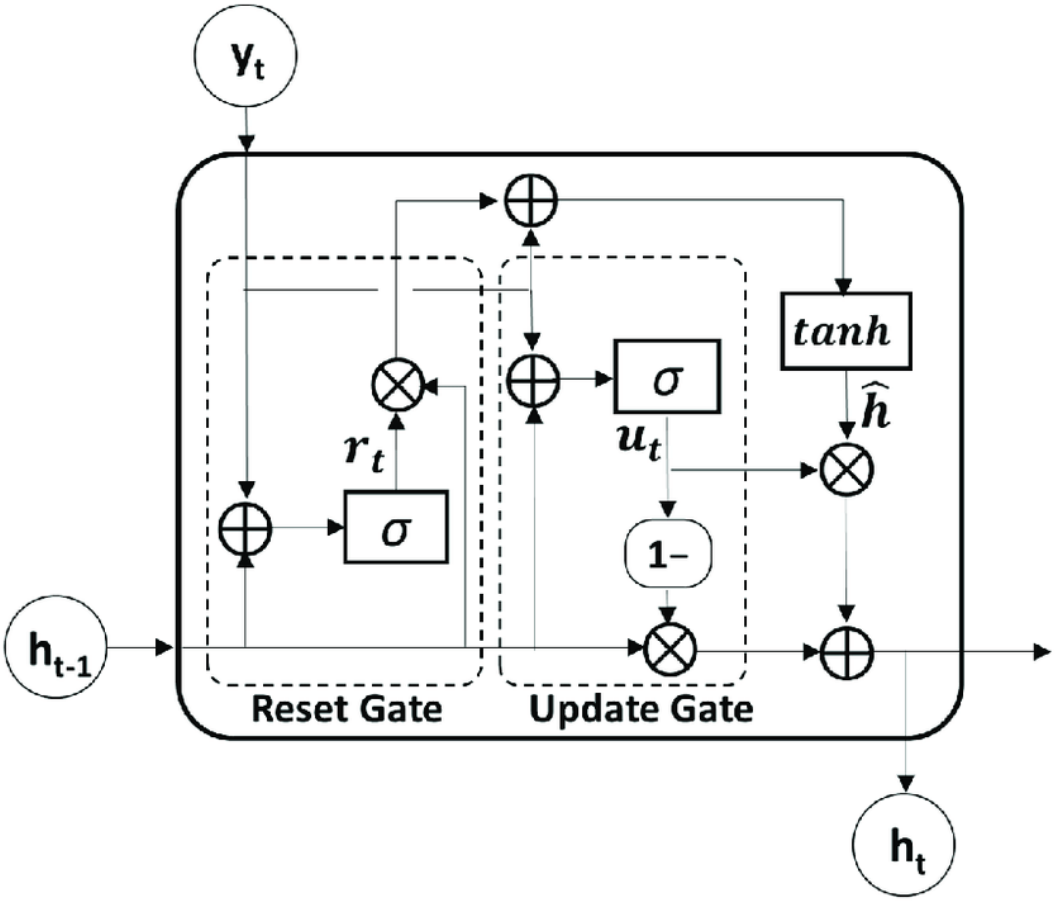


Figure 2.5: Structure of Gated Recurrent Units (Choi et al., 2022)

- The **candidate hidden state** $\tilde{h}_t$:

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b)$$

  It represents the new information being added to the memory.

- The **final hidden state** h_t at the current time step:

$$h_t = (1 - u_t) * h_{t-1} + u_t * \tilde{h}_t$$

  This is a combination of the old state and the new candidate state, adjusted by the update gate, with both the element-wise multiplication and addition operations involved

Here, σ denotes the sigmoid activation function, and $*$ denotes element-wise multiplication. The parameters W_r, W_u, W are weight matrices, b_r, b_u, b are bias vectors, h_{t-1} is the previous hidden state, x_t is the current input, and h_t is the current hidden state.

GRUs provide a balance between computational efficiency and the capability to handle sequences with long-term dependencies.

2.3.2 Convolutional neural networks

CNNs utilize convolutional layers, featuring filters that parse input data to identify features. These filters excel at detecting visual patterns in images and temporal trends in time series data. As they slide over the input, these filters understand key temporal characteristics, similar to how they detect textures and shapes in visual data (L. Liu and Si, 2022).

In time series analysis, CNNs can handle a variety of tasks, including but not limited to forecasting, anomaly detection, and feature extraction.

CNN Architecture for Time Series

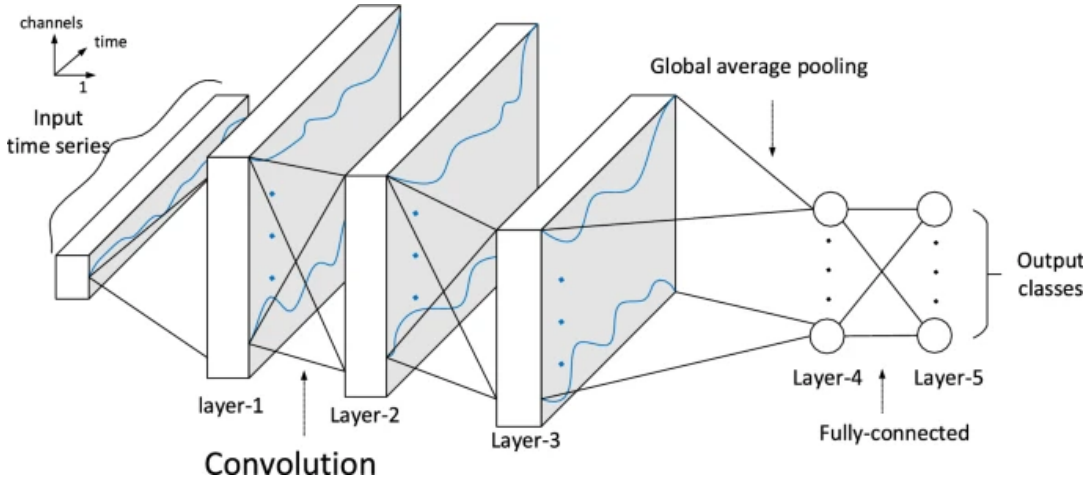


Figure 2.6: Basic 1D Convolutional neural network (L. Liu and Si, 2022)

Convolutional Layers:

These layers apply learned filters to the time series data, capturing local dependencies and extracting relevant features. In time series, these might correspond to short-term trends or repeating patterns (Shiri et al., 2023).

Activation Layers:

Typically, non-linear activation functions like ReLU (Rectified Linear Unit) follow the convolutional layers, introducing non-linearities to the model, which allow for more complex representations.

Pooling Layers:

Pooling (subsampling or downscaling) reduces the dimensionality of the data, which not only decreases computational load but also helps the model become more robust to the small variations in the time series data.

Fully Connected Layers:

Convolutional and pooling layers extract features in a neural network, while the fully connected layers perform high-level reasoning. In time series data, these layers interpret extracted features for the final prediction or classification.

2.3.3 Autoencoders

An autoencoder's architecture aims to break down and reconstruct input data accurately. Data compression is achieved through a bottleneck layer, smaller than both the input and output layers, creating a condensed code. The network's reconstruction is based on this code, utilizing two key components: the encoder and the decoder (Michelucci, 2022b).

Encoder: This part compresses input into a lower dimensional latent space.

Decoder: The decoder takes the encoded data and expands it back to the original input size. It aims to reconstruct the input data as accurately as possible from the encoded representation.

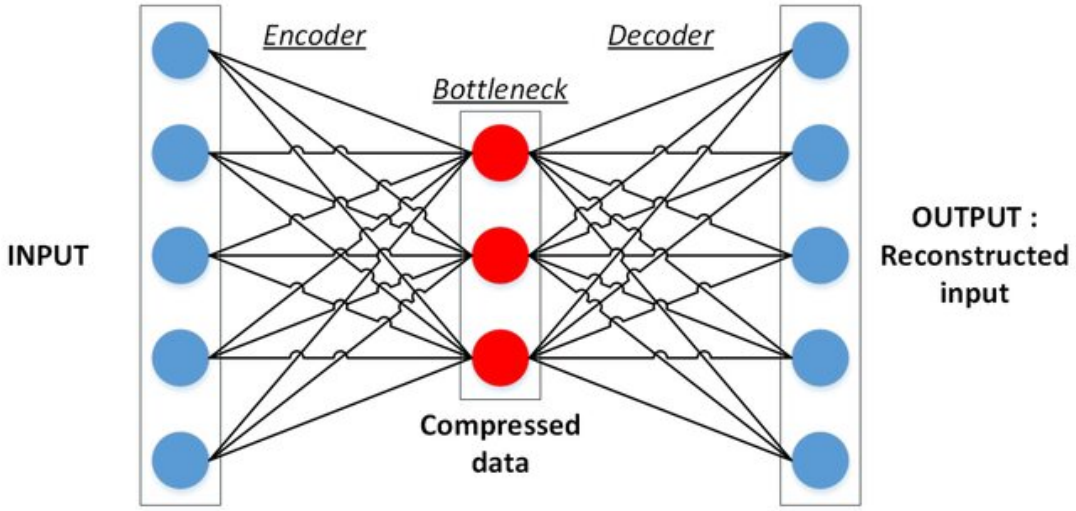


Figure 2.7: Structure of Autoencoder (Sublime and Kalinicheva, 2019)

The Autoencoder Process

The process involves the following steps (Michelucci, 2022b):

1. The encoder function, typically denoted as $h = f(x)$, compresses the input into a latent-space representation.
2. The decoder function, typically denoted as $r = g(h)$, aims to reconstruct the input from this representation.

The network is trained by minimizing the reconstruction error.

2.4 Federated Learning

Federated learning is an approach in machine learning where models are collaboratively trained across various decentralized devices or servers with local data, without data exchange. Introduced by (Konecnỳ et al., 2016) with Google, federated learning is particularly beneficial for maintaining privacy, securing data, and complying with local data regulations.

In a federated learning setup, the goal is to create a shared model by leveraging data distributed across multiple client devices while keeping the data localized. Instead of sending the data to the model, the model comes to the data (McMahan et al., 2017).

The federated learning process typically follows these steps (McMahan et al., 2017):

1. **Initialization:**
    - A global model is initialized on a global server.
2. **Local Training:**
    - The global model is sent to the clients (devices or local servers).
    - Each client trains the model on their local data.
3. **Local Updates:**
    - After training, the clients send their model updates, usually gradients or model weights, back to the global server.
4. **Aggregation:**
    - The global server aggregates these updates to create a new, updated global model.
5. **Model Distribution:**
    - The updated global model is sent back to the clients, and the process repeats.

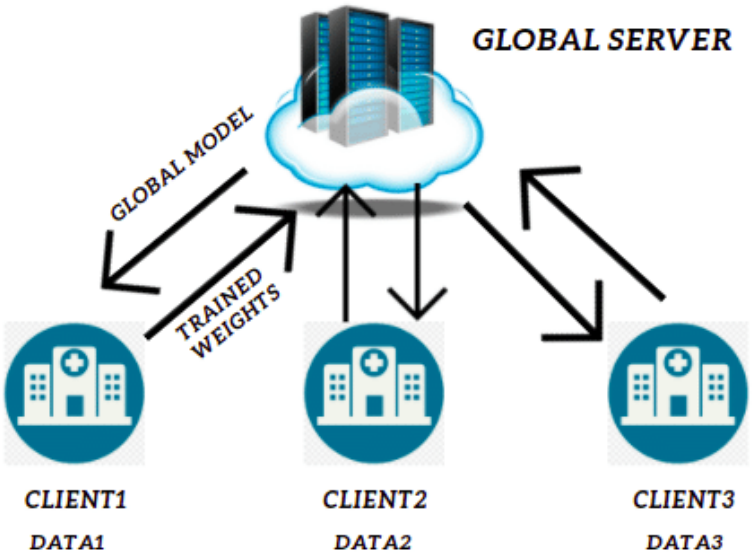


Figure 2.8: The Federated learning Process (Bhattacharya et al., 2022)

Let's formalize the Federated Learning (FL) process with some mathematical notation:

- w: The weights of the global model.
- w_i: The local model weights of the i-th client.
- $\mathcal{L}$: The loss function.

The objective in FL is to minimize the global loss function $\mathcal{L}(w)$, which is a weighted sum of the client's local loss functions:

$$\mathcal{L}(w) = \sum_{i=1}^{K} p_i \mathcal{L}_i(w_i)$$

where:

- K is the total number of clients.
- p_i is the relative size of the i-th client's dataset.
- $\mathcal{L}_i$ is the local loss function for the i-th client.

The global model weights are updated through aggregation:

$$w = \sum_{i=1}^{K} p_i w_i$$

where the local weights w_i are typically computed using local gradients.

Federated Averaging

Federated Averaging, commonly known as FedAvg, is a pivotal algorithm in federated learning. Devised by Brendan McMahan and the team at Google (McMahan et al., 2017), it aggregates models or updates across multiple clients within a federated network. Its simplicity and effectiveness have cemented FedAvg as a foundational algorithm in numerous federated learning implementations.

The main idea behind FedAvg is to average the updates (such as gradients or parameters) from multiple clients to update the global model.

Given K total clients, with k being a client participating in the current round, the FedAvg algorithm updates the global model weights w by averaging the submitted updates w_k from the clients:

$$w = \frac{1}{K} \sum_{k=1}^{K} w_k$$

Each client's update w_k is the result of local optimization, typically using stochastic gradient descent (SGD) or its variants on their local data:

$$w_k = w_k - \eta \nabla L_k(w_k)$$

where:

- η is the learning rate.
- $\nabla L_k(w_k)$ is the gradient of the local loss function to the local model weights.

Chapter 3

Literature Review

In this chapter, we discuss the topic of anomaly detection in time series mobile data. We also focus on the practical implementations and existing literature in this research area, with an emphasis on the use of federated learning.

The objective is to conduct a comprehensive analysis of various sources, encompassing empirical research, theoretical frameworks, and contemporary developments. This assessment aims to enhance understanding of the current state of the field.

3.1 Fundamentals of Time Series Data

Time series data is a type of data that is characterized by its sequential order. One primary use of this data is in predicting future events. For instance, Lim's work (B. Lim and Zohren, 2021) explores time series models for forecasting. Additionally, identifying anomalies is another important use of time series data. Blazquez provides a comprehensive review (Blázquez-García et al., 2021) of methods for anomaly detection. These tasks, among others, demonstrate the significance of time series data in fields like mobile network data analysis.

Time series data is inherently ordered in time and can be discrete or continuous. A fundamental method for its analysis is the Autoregressive Integrated Moving Average model. This technique is extensively explained by Box and others in their publication "Time Series Analysis: Forecasting and Control" (Box et al., 2015). This is often regarded as a cornerstone in the field of time series forecasting.

The model combines autoregression, integration, and moving averages to enable sophisticated forecasting (Hyndman and Athanasopoulos, 2018). Since its introduction, ARIMA has been widely used in various fields and has motivated a lot of research and applications in this field (Benvenuto et al., 2020; Wagner and Cleland, 2023; Kaur et al., 2023). The temporal distribution of time series data, to its occurrence in time, is the foundation of many analytical frameworks and decision-making

processes (Wagner and Cleland, 2023).

3.1.1 Univariate and Multivariate time series

Time series data representations can be classified majorly into two, *Univariate* and *Multivariate*, with (Audibert et al., 2020) properly describing the difference between these two classifications of time series. A univariate time series is a collection of individual data points recorded at specific times, representing observations of a single process. On the other hand, multivariate time series capture observations of multiple variables at each time point, providing a more comprehensive view by tracking several variables simultaneously.

$$Univariate : \{x_1, x_2, \ldots, x_T\}$$

$$Multivariate : \{\mathbf{xt}_1, \mathbf{xt}_2, \ldots, \mathbf{xt}_m\}$$

3.1.2 Time Series in Mobile Network Data

Time series data in mobile networks introduce amplified challenges. These challenges are intensified by the need for consistent availability, stringent security, and careful privacy maintenance (Feamster and Rexford, 2017). Mobile data are not only voluminous but are also produced and gathered rapidly, necessitating timely analysis. The data vary from call logs to network traffic metrics, often referred to as Key Performance Indicators(KPIs). Analysis complexity increases as outcomes and precision greatly depend on the chosen KPIs (Shakir et al., 2023).

Several research efforts aim to enhance time series analysis comprehensiveness in mobile network data and to address inherent data structure issues, thus boosting accuracy and efficacy.

One study, utilizing a dataset from a Portuguese mobile network operator, focuses on predicting traffic volumes for downloads and uploads within hourly intervals. It highlights that the Seasonal Autoregressive Integrated Moving Average model is more effective for forecasting download patterns, while the Holt-Winters model excels in upload traffic predictions (Kochetkova et al., 2023).

Another study employs a hybrid model merging Double Seasonal ARIMA with Long Short-Term Memory networks, using data from 739 base stations. This model is enhanced with K-means clustering, handling high volumes typical of network data, and improves the detection of double-seasonality, non-linear trends, and spatial dependencies (Shawel et al., 2020).

Researchers are also exploring frameworks to manage the complexities of big data, like mobile network data, involving time series monitoring, forecasting, and anomaly detection. These frameworks can be tailored to the specific challenges in analyzing mobile data (Almeida et al., 2023).

3.1.3 Anomaly Detection in Mobile Network Data

Anomalies, or outliers, are observations that significantly deviate from a dataset's normal distribution or pattern. These irregularities can manifest in various forms, such as isolated data points, clustered anomalies, or unusual patterns within specific dataset regions (Samariya and Thakkar, 2023). Anomaly detection is crucial across diverse domains, serving as a key process for identifying deviations from expected patterns in data.

In the context of mobile networks, analyzing time series data for anomalies presents distinct challenges due to the data's high-dimensional nature. Machine learning techniques are instrumental in discerning temporal dependencies and spatial correlations within historical data (Tian et al., 2023). These techniques help to learn the expected distribution of the data which serves as a baseline for identifying anomalies.

Anomalies are deviations from this norm and can potentially lead to signal network issues, fraudulent activities, or unforeseen events. By accurately detecting these anomalies, we can take appropriate actions to prevent or resolve any issues that may arise.

There are three main types of anomaly detection methods: supervised, semi-supervised, and unsupervised. This review provides an analysis of each methodology, highlighting key contributions within the field.

Supervised Anomaly Detection

Supervised learning is an effective method for detecting anomalies in mobile networks. It works best when labeled data is available, containing both normal and anomalous behavior. The training process is flexible and allows for high-accuracy models that can easily detect known anomalies within the training dataset. Moreover, the interpretability of this method helps to identify the root causes behind the identified anomalies.

The study by Ahmed and Pathan, 2019 examines deep learning and various supervised learning techniques in detecting collective anomalies. Collective anomalies are identified when a cluster of data points diverges from the norm, yet each instance might not appear anomalous. The research demonstrates deep learning's capacity to detect such anomalies with about a 97 percent recall rate, based on tests using the UNSW-NB15 and KDD Cup 1999 datasets (Moustafa and Slay, 2015; Tavallaee et al., 2009).

The paper by Ahasan et al., 2022 investigates anomaly detection in mobile networks using supervised learning, utilizing classification algorithms to parse KPI data. This research targets key performance indicators, such as call setup success, handover success rates, and radio resource utilization, to detect anomalies in mobile networks. Early identification is essential for maintaining network integrity and enhancing user experience.

The study assesses various classification methods, notably Support Vector Machines, XGBoost, and

Random Forest, to identify the most effective approach. To tackle the issue of data imbalance, the authors incorporate the Synthetic Minority Oversampling Technique. The efficacy of these models is then measured using a real-world mobile network dataset with pre-labeled KPIs. Evaluation metrics used include accuracy, precision, recall, and the F1-score, where the models demonstrate high accuracy. the paper highlights the need for sufficient labeled data and attention to potential limitations like imbalanced datasets and interoperability.

Supervised learning encounters significant challenges in dynamic environments like mobile networks. Firstly, its heavy reliance on large amounts of labeled data becomes even more cumbersome with high-dimensional mobile network data. Secondly, the ever-evolving threat landscape and data distribution shift over time hinder its ability to scale well to new and unknown threats. This becomes critical in mobile networks demanding close-to-real-time prediction, where delays in adaptation can leave them vulnerable.

Semi-Supervised Anomaly Detection

Semi-supervised learning is a promising approach that has emerged as an alternative to traditional supervised learning. This method trains machine learning models by utilizing both labeled and unlabeled data. Labeled data provides initial guidance to the model, while the abundance of unlabeled data helps to refine the model and enhance its ability to generalize to new scenarios. By utilizing both sources effectively, semi-supervised learning can improve the accuracy and efficiency of machine learning models by removing the problem of labeling data and building models that are adaptable (Zhang et al., 2023).

The research work by Zhang et al., 2023 proposes a new algorithm to address the challenges of network anomaly detection. The algorithm uses semi-supervised learning and adaptive multiclass balancing to overcome the issues of label inadequacy and class imbalance. The algorithm employs rotation decision trees as component classifiers, which ensures diversity and enhances generalizability. The multiclass split balancing strategy is used to divide labeled data into balanced subsets for diverse classifier training. Additionally, the adaptive confidence threshold function is used to select reliable pseudo-labeled data during updates. This effectively addresses class imbalance, a huge problem in supervised learning, by assigning class-specific confidence thresholds. By combining these strategies, the proposed algorithm aims to improve detection performance in environments with limited labeled data and imbalanced classes. The research shows that the proposed algorithm outperforms state-of-the-art methods of precision, recall, and F1.

A study by Pelati et al., 2022 addresses the issue of high dimensionality of mobile data using semi-supervised learning. Their approach aims to identify unusual traffic patterns caused by events such as flash crowds in metropolitan areas, without requiring prior knowledge of the anomalies. The authors use real traffic data from unencrypted LTE Physical Downlink Control Channels (PDCCH) collected in major Spanish cities. The methodology's use of unsupervised learning (K-means clustering) for pre-processing to filter out outliers aligns with the emphasis on managing high-volume data. LSTM networks are used in two configurations: predictive and reconstructive, for forecasting future traffic values and reconstructing input samples, respectively. The proposed method outper-

forms traditional anomaly detection algorithms, offering a high degree of accuracy in predicting anomalous behavior regardless of its cause.

In a recent study published by Meng et al., 2021, introduces a new framework called SemiADC for detecting anomalies in communication networks. The framework employs a semi-supervised learning approach and addresses the issues of inaccurate labels in anomaly detection. To overcome these challenges, SemiADC utilizes Generative Adversarial Networks (GANs) and self-learning processes. The framework learns the distribution of features of normal nodes with regularization from abnormal ones, iteratively cleans the data, and extracts nodes with inaccurate labels to improve the accuracy of anomaly detection. The experimental results indicate that SemiADC outperforms the existing state-of-the-art methods without requiring adequate and accurate supervision. The framework has been evaluated on real-world datasets, demonstrating its effectiveness in accurately identifying anomalies in dynamic communication networks.

Semi-supervised learning often relies on pseudo labels for unlabeled data, which are generated by the model itself. If these pseudo labels are inaccurate, they can lead to the propagation of errors during the training process, degrading the performance of the model. The design of semi-supervised also algorithms often requires careful tuning and a deeper understanding of the underlying data structure, which can be a complex and time-consuming process (Van Engelen and Hoos, 2020).

Unsupervised Learning: A Growing Force In Anomaly Detection

Unsupervised learning is an essential methodology for detecting anomalies within the vast and complex datasets of mobile networks. This approach does not rely on pre-labeled data, making it advantageous in scenarios where labeling is impractical due to the volume of data or the ambiguous nature of what constitutes an anomaly (S. Wang et al., 2021).

Su et al., 2019 serves as a baseline method for multiple research within multivariate time series anomaly detection. In this paper, they propose a novel approach for anomaly detection in complex multivariate time series. The paper introduces OmniAnomaly, a model that combines stochastic variables and recurrent neural networks to capture the intricate temporal dependencies and variability in data without requiring labeled data. It utilizes a planar normalizing flow for better reconstruction of input data, facilitating accurate anomaly identification. The paper's experiments, conducted on real-world datasets, showcase OmniAnomaly's superior performance in detecting anomalies within time series data compared to existing methods at the time, which has led to more research in this field.

Unsupervised Anomaly Detection on Multivariate Time Series (USAD) (Audibert et al., 2020). This method uses autoencoders in an adversarial training framework to detect anomalies in multivariate time series data. The model's effectiveness was evaluated on five public and private datasets from Orange. Results showed that USAD has high training speed, robustness, and superior performance compared to state-of-the-art methods. The architecture of USAD allows it to learn normal patterns and detect anomalies quickly and reliably, making it a promising solution for real-world applications that require fast and dependable anomaly detection. However, it is unsure whether the model could

handle the potential challenges of adapting to highly dynamic or non-stationary data environments typical in mobile networks.

In their recent study, Shen et al., 2020 proposes a new approach for detecting anomalies in time series data called the Temporal Hierarchical One-Class (THOC) network. Unlike traditional supervised methods, THOC works in an unsupervised manner, making it practical for situations where labeled anomalies are scarce. The authors tested this model on real-world data and used the widely recognized KDD Cup 99 dataset for intrusion detection, further adding to its significance. The THOC network uses an extended recurrent neural network (RNN) with skip connections to capture multiscale temporal features. These features are then fused through hierarchical clustering using a Multiscale Support Vector Data Description (MVDD) approach. This approach measures differences between features and centers, enabling THOC to represent complex behaviors in time series data. The authors conducted extensive experiments to demonstrate the superiority of THOC over recent baselines. Therefore, THOC is a promising advancement in unsupervised anomaly detection. The authors integrated both neural networks using RNN to capture the temporal dynamics in the data and cluster the spatial relationships.

In a recent study, Wei et al., 2023 focuses on detecting Distributed Denial-of-service (DDoS) attacks within networks. These attacks aim to disrupt regular traffic by overwhelming a targeted server, service, or network with malicious traffic. The researchers propose an unsupervised anomaly detection method using an LSTM-Autoencoder (LSTM-AE) model. The LSTM networks capture long and short-term correlations in time-series data, while the AE learns compact representations of data. The LSTM-AE reconstructs multivariate time-series (MTS) data and then calculates the reconstruction error for each sample across all time-series sequences, A high reconstruction error indicates an anomaly, such as a DDoS attack. The researchers utilize the CICDDoS2019 dataset (Sharafaldin et al., 2019), widely used for DDoS attack detection and classification. This dataset contains realistic samples of malicious network traffic. The LSTM-AE model learns subtle subpattern differences between attacks and benign traffic flows. It minimizes reconstructed anomalous traffic to achieve a lower reconstruction error. For reflection-based DDoS attacks (such as DNS, LDAP, and SNMP), LSTM-AE outperforms other state-of-the-art methods, achieving an accuracy of over 99 percent, however, LSTM-AE combines both LSTM networks and autoencoders, resulting in a complex architecture which makes training such a hybrid model and multiple layers in the LSTM challenging.

In their research paper, González et al., 2023 proposes an approach called Dilated Convolutional - Variational Auto Encoder (DC-VAE) that combines the power of convolutional neural networks (CNNs) and variational autoencoders (VAEs). The DC-VAE approach uses dilated convolutions to capture both short and long-term phenomena in the data. VAE is used for probabilistic modeling to enhance generalization. DC-VAE detects anomalies in MTS data through a single model, obtaining the temporal information without sacrificing computational and memory resources. The researchers evaluate the proposed method on the TELCO Telecommunication-networks dataset, which is a large-scale, multi-dimensional network monitoring dataset. Anomalous events in this dataset were manually labeled by experts over seven months. The results show that DC-VAE outperformed traditional time-series anomaly detectors from signal processing and machine learning domains.

Tian et al., 2023 proposes a novel approach called STADN (Anomaly detection using spatial and temporal information) The authors try to address the lack of labeled data for supervised machine learning due to the rarity of abnormal instances by employing an unsupervised approach but also to adequately utilize the spatial and temporal dependencies among multiple variables within time series data. STADN applies an attention mechanism by modeling the relationship graph between variables using a graph attention network to capture the spatial dependency between variables. It employs an LSTM network to get the temporal dependency of time series data before predicting the future behavior of each sensor by combining historical behavior with that of its neighbors. Anomalies are detected and located based on prediction errors and the model improves its ability to discriminate between anomaly and regularity by reconstructing prediction errors. The proposed STADN was evaluated on two real-world datasets and outperformed state-of-the-art anomaly detection performance. While attention mechanisms enhance the model's ability to capture relevant features, they do not guarantee generalization to novel or unseen anomalies.

The paper Hua et al., 2023 proposes a method for detecting anomalies in wireless base stations (WBS) in large-scale networks. It is called GenAD; General Unsupervised Anomaly Detection Using Multivariate Time Series. The method uses attention mechanisms to identify spatial relationships and temporal features. GenAD uses multi-correlation attention and time-series attention to learn complex correlations and temporal patterns. The model is pre-trained on large-scale WBS data, allowing it to be easily transferred to specific stations with minimal additional training data. Once trained, it analyzes MTS data from each WBS. If a specific metric, such as wireless connection rate, deviates significantly from expected behavior, GenAD flags it as an anomaly. The model's attention mechanisms help it identify anomalous patterns by weighing different time steps and features. GenAD's performance remains competitive, even when trained with only 10% of the data. Attention mechanisms allow the model to focus on relevant parts of the input time series, and the attention weights provide insights into which time steps or features contribute most to anomaly detection. This transparency helps diagnose network issues and improve system reliability.

The paper by Matar et al., 2023 proposes an innovative approach to detect anomalies in wireless sensor networks (WSNs) data streams. The authors introduce a combination of multi-head attention and a Bi-LSTM (Bidirectional Long Short-Term Memory) architecture to address the challenge of detecting anomalies in WSNs data streams. Multi-head attention is a deep learning mechanism that improves the model's ability to focus on different parts of the input sequence. In this approach, the input sequence is transformed into multiple representations, or "heads," each of which attends to different parts of the sequence. The attention mechanism is then applied to each head separately, and the results are concatenated to form the final output. Unlike traditional methods that model individual sensor time series independently. This approach concurrently models the time series of multiple sensors. By considering potential latent interactions among sensors, the accuracy of anomaly detection is improved. Moreover, the proposed approach does not require labeled data, making it suitable for real-world scenarios where labeling large streams of data from heterogeneous sensors is challenging and time-consuming. The authors validate the effectiveness and robustness of their approach using a real-world WSN dataset. The proposed method outperforms traditional deep learning approaches in terms of anomaly detection accuracy.

Despite their benefits, unsupervised learning models encounter two significant challenges in mobile

networks. First, they require a considerable amount of computational power caused by the high dimensionality of mobile network data. Second, they struggle to cope with the constantly changing and widely dispersed nature of mobile network data. These challenges emphasize the necessity for new approaches that can take advantage of the distributed nature of mobile data, preserve user data privacy, and consume less computational power.

3.1.4 Distributed Learning

Federated Learning An Emerging Solution

Federated learning is a solution to the challenges posed by traditional centralized learning models. It enables a decentralized learning process where multiple mobile devices learn a shared prediction model collaboratively while keeping all the training data on the device. This approach ensures that the ability to perform model training is separate from the need to store data in the cloud, addressing privacy concerns. Additionally, this model leverages the distributed nature of mobile networks to enhance anomaly detection capabilities (McMahan et al., 2017).

By utilizing a decentralized approach, federated learning has proven to overcome the limitations of traditional machine learning models. Numerous research studies across different domains support this statement (B. Liu et al., 2023; Pfitzner et al., 2021; Shaheen et al., 2022). In this discussion, we will explore these benefits using specific research examples as evidence of Federated learning's efficacy.

Privacy Preservation

Federated learning is recognized as a key method for processing sensitive data in the context of mobile networks, safeguarding user privacy, and addressing concerns about sharing proprietary data for research.

A study by Qian et al., 2019 demonstrates the application of distributed federated learning for optimizing service placement decisions in mobile edge clouds, ensuring privacy is not compromised, and underscoring the limitations of centralized learning models.

Within 5G mobile networks, Isaksson and Norrman, 2020's research proposes an innovative federation of learning with secure Multi-Party Computation protocols to protect individual data privacy while optimizing machine learning performance, reflecting the critical balance between data confidentiality and the imperatives of network data sharing. These developments highlight the essential role of privacy preservation in the evolution of mobile network technologies.

Bandwidth Efficiency

Federated learning highlights the practicality of local data processing on devices in mobile networks, where only model updates, rather than the data itself, are communicated, conservatively utilizing bandwidth.

The research by Tuli et al., 2022 highlights innovative methods for optimizing computation offloading, traditionally a high bandwidth-consuming process and a challenge for bandwidth-constrained edge devices. By engaging federated learning, computational tasks are distributed among edge devices, which aggregate their updates, thereby circumventing bandwidth limitations while preserving performance levels.

Real-time Adaptation

Real-time prediction is an ideal goal but achieving perfect accuracy can be difficult due to the dynamic nature of mobile networks and limitations of the models. In practice, it is more important to focus on timely detection and minimizing false positives. The article by Kang et al., 2023 discusses the need for real-time prediction in the context of the Internet of Vehicles (IoV). The authors point out the challenges of traditional centralized machine-learning approaches, which face issues related to latency, privacy, and scalability. To address these challenges, they proposed a Federated Learning-based modeling algorithm that is well-suited for real-time prediction in dynamic and distributed edge networks. The decentralized training approach of Federated Learning aligns with the nature of edge networks and enables the algorithm to overcome the challenges faced by traditional centralized machine learning approaches.

Anomaly detection Using Federated Learning

In their recent study, Sater and Hamza, 2021 introduces a federated learning model that uses stacked LSTM networks to identify anomalies in smart buildings. The model employs a multi-task learning approach to solve multiple tasks simultaneously, improving the accuracy of both energy usage prediction and anomaly detection using IoT sensor data while preserving privacy. The researchers used the Sensors Event Log Dataset for their study and found that their model outperformed centralized models regarding training speed and prediction performance. The results demonstrated the effectiveness of their approach in real-world scenarios, specifically in General Electric Current's smart building dataset. This method not only reduces the overall costs of training but also maintains prediction accuracy, showing a significant advance in privacy-preserving anomaly detection in IoT environments.

In their research, Sheikhi and Kostakos, 2023 propose an approach for detecting DDoS attacks in 5G core networks using unsupervised federated learning. The study highlights the importance of implementing advanced security measures to combat DDoS attacks that target the GTP protocol. By utilizing the collective intelligence of multiple devices, the model efficiently and privately identified DDoS attacks. The authors also introduced a 5G testbed architecture that simulates complex public networks, which is ideal for evaluating AI-based security applications, while implementing an autoencoder model with Fedavg for server-side aggregation. The results demonstrated the model's effectiveness in detecting DDoS attacks while maintaining data privacy. This emphasizes the potential of federated learning to enhance the security of 5G networks.

In their recent paper, Marfo et al., 2022 propose a new approach called hierarchical federated learning (HFL) for detecting network anomalies, with a specific focus on DDoS attacks in 5G networks.

The HFL approach incorporates edge servers between clients and a global server, which helps to reduce the computational load on the global server, making the overall model more efficient and scalable. The researchers validate their approach using the UNSW-NB15 dataset (Moustafa and Slay, 2015) and demonstrate that it outperforms traditional centralized models in terms of accuracy and training efficiency. This study highlights the potential of federated learning in enhancing cybersecurity measures in network environments. However, while the hierarchical structure introduces efficiencies in model training and data aggregation, the architecture's scalability is dependent on the number of edge servers and their ability to manage client communications effectively.

The paper Vucovich et al., 2023 introduces a solution for convergence issues in federated learning through the use of a Min-Max Scalar and a sampling technique called FedSam. This approach was tested across various setups, including a three-party federated model that utilized datasets from NCC-DC National Collegiate Cyber Defense Competition, 2024, CIC-IDS2017, and CIC-IDS2018 Sharafaldin et al., 2018. The Min-Max Scalar algorithm works by initializing a scalar with random minimum and maximum values for each feature used in model training. This scalar is shared among clients sequentially, allowing each to adjust the scalar based on their data's bounds. The result is a common Min-Max Scalar that reflects the global data range. FedSam is an advanced federated aggregation strategy that combines aspects of Mini-Batch and Multi-Epoch FedAvg strategies to balance updates from all client nodes equally. This approach aims to mitigate the influence of data imbalances on the global model, promoting a more equitable and efficient learning process. These techniques represent significant advancements in federated learning, particularly in contexts where data distribution varies widely across participants.

In their study, Zhao et al., 2020 propose an approach to tackle the issue of data scarcity in deep learning-based network anomaly detection methods. Their method involves the combination of federated learning and transfer learning. In the federated learning stage, participants collaborate and share knowledge without revealing their raw data. The global model is then aggregated from the local models, preserving data privacy as the data remains on local devices. In the transfer learning stage, the global model is fine-tuned using labeled data from the UNSW-NB15 dataset Moustafa and Slay, 2015. This unique integration of federated learning and transfer learning provides a promising solution for enhancing network security, particularly in scenarios where data availability is limited.

Challenges in Federated Learning

Federated Learning is gaining popularity due to its privacy-preserving nature and its ability to use data from various sources. However, there are still several challenges that hinder its widespread adoption.

Data Heterogeneity

Federated learning increases complexity due to data heterogeneity, where devices or servers train models with their own distinct datasets leading to variations in distribution, features, and volume. This is unlike centralised machine learning's uniform data processing. Adapting to these has brought

about different strategies, as seen in recent studies. One study by Dai et al., 2023 introduces FedNH, a method that incorporates class semantics to enhance class prototypes and address class imbalance, achieving stability and better personalization in cross-device federated learning. Another study by de Luca et al., 2022 adapts domain generalization techniques to federated settings, utilizing data augmentation for heightened accuracy across diverse clients, even enhancing basic Federated Averaging algorithms.

Exploring preprocessing and aggregation methods, in a research by Perifanis et al., 2023 they recommend techniques like global scaling and advanced aggregation algorithms to manage the nuances of non-iid data effectively. These strategies can substantially bolster federated learning models amid the common challenges of data distribution skew, quantity skew, and temporal skew, providing a pathway to more robust and equitable machine learning in decentralized environments.

Privacy Preservation

Federated learning, while improving data privacy by decentralizing machine learning model training to user devices, faces privacy threats such as inference attacks, model inversion attacks, membership inference attacks, and property inference attacks which aim to extract sensitive details from the model updates (P. Liu et al., 2022). There has been some research into solving this problem

One study Nguyen et al., 2022 presents a method that relies on quantization and noise addition to the federated learning updates to protect private data in Mobile Edge Computing settings. This approach balances the need for privacy with communication resource optimization, ensuring model performance within privacy and network constraints.

Another research Mandal and Gong, 2019 proposes a privacy-enhancing system specific to linear and logistic regression models in federated learning. The system utilizes additive homomorphic encryption and secure aggregation to guard both user data and the server's model, proving effective even when users drop out of the learning process.

Thus, while vulnerabilities exist, advances in protecting confidentiality and preserving privacy in federated learning are promising, striking a balance between user privacy and the demands of mobile network applications.

Scalability

Efficiently managing federated learning as the number of devices grows while optimizing resource distribution is a critical challenge. Research in this area has led to various innovative solutions that aim to improve performance and learning efficiency without overloading the network or its participants (Zhou et al., 2023).

Ren et al., 2021 propose a framework designed to minimize system and learning costs in mobile edge computing environments and optimize bandwidth, computing frequency, power allocation, and subcarrier assignment. This approach employs problem decomposition, convex optimization, and algorithms like the Hungarian method to significantly enhance the outcomes of federated learning.

The Edge-Cloud Collaborative Knowledge Transfer Framework by Li et al., 2023 innovatively combines the strengths of federated learning at the edge with centralized learning in the cloud. This framework adapts to model differences, supports asynchronous training, and minimizes data transmission, improving federated learning's scalability and efficiency across a multitude of devices.

In the context of edge computing, Z. Wang et al., 2023 propose solutions to optimizing performance on resource-constrained edge servers include two algorithms based on the Alternating Direction Method of Multipliers, designed for both homogeneous and heterogeneous environments. These algorithms address the complex, multi-constraint optimization required for effective resource allocation.

Collectively, these contributions showcase the progress being made in enhancing the scalability and efficiency of federated learning, with a focus on intelligent resource allocation and leveraging both edge and cloud computing paradigms.

3.1.5 Is Split Learning A Better Solution?

Split learning was first introduced by Gupta and Raskar, 2018, This work splits a model between devices and a server. Devices process data and send intermediate results to the server for training completion. This method reduces data transfer, enhancing privacy and efficiency. It can be especially useful for mobile networks with limited resources, offering a scalable, privacy-focused solution for tasks like anomaly detection. In tests using the CIFAR 10 and CIFAR 100 datasets (Alex, 2009), split learning proved superior to federated learning. It achieved better accuracy convergence and required less computation on the client side (Vepakomma et al., 2018).

In Z. Lin et al., 2023, an adaptation of split learning in mobile networks in exploring the integration of split learning with 6G edge networks to address the limitations of federated learning in resource-constrained environments is proposed. A comprehensive review of Split Learning is presented, including its advantages for privacy enhancement and reduced resource demands on IoT devices. The paper discusses the architecture for split-edge learning, resource-efficient Split Learning frameworks, and resource management strategies, emphasizing the potential of Split learning to become a key AI technology in the 6G era. It identifies critical design challenges and future research directions, including efficient split learning design, resource management, and addressing open problems such as convergence analysis and asynchronous learning.

Combining Split And Federated Learning

Split-Federated learning proposed by Thapa et al., 2022 is an innovative approach that merges the strengths of both Federated Learning and Split Learning. By dividing the network architecture between clients and a server, similar to Split Learning, and adopting Federated Learning's parallel model update strategy, Split-Federated learning enhances privacy and computational efficiency. Empirical tests suggest it maintains communication efficiency and test accuracy on par with Split

Learning while reducing computation time, making it a viable option for real-time anomaly detection in mobile networks.

Another stride in this field is the introduction of a framework, FedSL by Abedi and Khan, 2023, which is tailored for training Recurrent Neural Networks on distributed sequential data, combining federated learning with split learning. This method is adept at sequential data analysis, ensuring privacy since it avoids the sharing of raw data or full model parameters. FedSL shows improved privacy, accuracy, and communication efficiency over existing methods, proving advantageous in areas like time series and natural language processing.

These contributions reflect the continual evolution of hybrid distributed learning models that prioritize privacy preservation and computational resource optimization, especially suitable for mobile and sequential data environments.

Split Learning Integrated With Other Learning

The SALAD by Nixon et al., 2023 method combines autoencoders and active learning to enhance anomaly detection in network data streams, aiming to minimize labeling expenses, speed up training and adjustment, as well as improve detection performance. It utilizes split learning, where the autoencoder's client-side up to the split layer processes data without exposing server-side layers, ensuring privacy. Rather than relying on a static dataset, SALAD's active learning selects data points for experts to label, which then refine the client-side autoencoder. This leads to more effective adaptation to evolving data and reduces the necessity for extensive human annotation. Testing on various datasets showed that SALAD surpasses conventional anomaly detection techniques with fewer labeling requirements, offering a potent approach for efficient and real-time network threat identification.

Chapter 4

Methodology

This Chapter describes the methodological framework adopted to investigate anomaly detection in mobile networks using machine learning techniques while implementing federated learning. The methodology presented here has been designed to address research questions systematically and provide a clear road map from data collection to model evaluation.

4.0.1 Research Approach Overview

The research methodology applied in this study is divided into distinct phases, each critical for navigating the intricacies of applying machine learning to our mobile data for anomaly detection. These phases are:

Data Collection and Pre-processing: This phase involves the acquisition of the mobile network data and its subsequent preparation, to ensure it is suitable for further analysis.

Machine Learning Model Development: The focus here is on selecting and training appropriate machine learning models for anomaly detection as well as the implementation of federated learning. The rationale behind model choice and the training process will be elaborated.

4.1 Data Collection And Preprocessing

This section outlines the procedures implemented for data collection. Furthermore, we elaborate on the data source and the initial investigations performed by the researchers, which facilitated the data's availability. This discussion also highlights the relevance and appropriateness of the selected data for this study.

4.1.1 Data Collection

The Nornet Edge (NNE) platform is a robust infrastructure with over 400 measurement nodes in Norway. It conducts extensive experiments in mobile broadband networks and enables long-term assessment and comparison of quality and performance across different network operators on a national scale. The platform also includes a backend system that simplifies research by deploying experiments, managing nodes, and collecting data (Kvalbein et al., 2014).

Figure 4.1: A measurement box with an NNE node and a power source (Ahmed and Pathan, 2019)

The data collection methodology in this study capitalizes on the distinctive features of the Nornet Edge (NNE) platform, which supports simultaneous experimentation across various mobile broadband (MBB) connections. This capability is essential for capturing comprehensive context information related to the experiments. Utilizing measurement nodes equipped with Linux operating systems and linked to multiple MBB providers, the study gathers an extensive array of data across a spectrum of conditions, ranging from urban centers to isolated areas. Such an approach is instrumental in conducting an in-depth analysis of mobile data traits, ensuring the representativeness of the collected data across the diverse network environments found throughout Norway.

The distributed nature of this platform facilitates the incorporation of a federated learning framework into this study. By providing detailed insights into location-specific data, the platform enables the identification of geographically correlated data trends and anomalies.

The NNE platform's backend system plays a pivotal role in experiment deployment, node manage-

ment, and data processing. This infrastructure's proficiency in collecting and analyzing data from varied settings and network scenarios is vital for discerning historical trends in mobile network data as they relate to node locations. For this investigation, we select one node from Tromsø, Geilo, Bud, and Oslo as our focal points. These locations are specifically chosen to develop a model that generalizes across Norway's different locales.

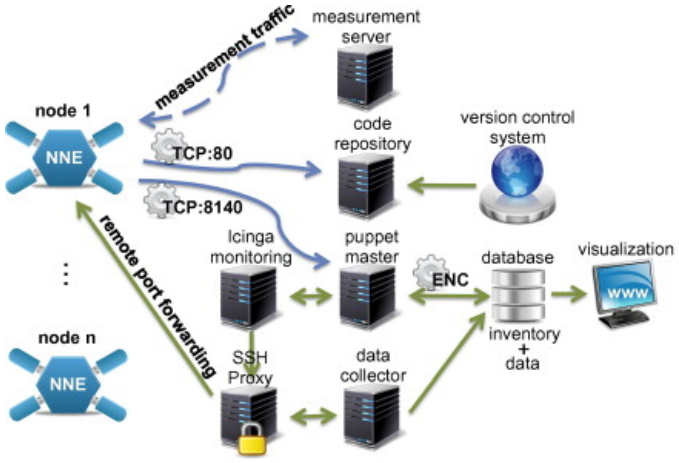


Figure 4.2: NNE Backend structure (Kvalbein et al., 2014)

Preliminary Research

Identification of Key Performance Indicators (KPIs)

The initial phase of the research involves identifying the most pertinent measurements for detecting anomalies within mobile network performance. Given the study's focus on network anomalies, it is important to select Key Performance Indicators (KPIs) that accurately reflect the network's performance state. These KPIs needed to account for potential fluctuations that could either stem from the condition of the measurement node itself or signify underlying issues within the network's performance.

Among the various KPIs considered, Round Trip Time (RTT) is selected as a principal metric due to its robust indication of network performance. RTT measures the time required for a signal to travel to a destination and back, which is an indicator of network speed and efficiency. Notably, RTT's fluctuations highlight potential network security issues, such as Denial of Service (DoS) or Distributed Denial of Service (DDoS) attacks, making it a critical KPI for detecting anomalies within the network (Goudru et al., 2021; Martinez et al., 2023; Sengupta et al., 2022).

To complement the analysis, we also select metrics such as Reference Signal Received Quality (RSRQ), Reference Signal Received Power (RSRP) (Afroz et al., 2015; El-Saleh et al., 2022), and Received Signal Strength Indicator (RSSI) (Santana et al., 2017; Sarsodia et al., 2021). These metrics are instrumental in describing the strength and quality of the network signal, which are expected to correlate significantly with RTT measurements. A strong and stable signal, as indicated

by favorable RSRQ, RSSI, and RSRP values, typically contributes to lower RTT values, signifying efficient network performance. Conversely, weaker signal strengths may lead to increased RTT, indicating potential performance issues or areas with poor network coverage.

Further data elements, including the node's mode of operation, LTE frequency, band, information service, and network ID, are collected to provide a holistic view of the network's operational state. These details not only aid in distinguishing the type of network measurement but also in identifying the service provider, thus offering comprehensive insights into the network's performance dynamics.

Data Collection Time-frame And Scope

To ensure the most accurate analysis of the network's performance and security posture, the data collection period for the study is strategically chosen to be from November 1st to December 31st, 2023. These two months are selected to capture the most current state of the network across the targeted areas. This approach sets the study to reflect the prevailing conditions.

Data Structure:

Packet Loss And RTT Data

To conduct a detailed examination of network anomalies, the study relies on two main types of data: packet loss data we obtain from UDP ping experiments and comprehensive metadata records. The packet loss data, which plays a crucial role in comprehending the network's strength and effectiveness, comprises the following essential components:

- Measurement Timestamp (`ts`): The precise moment of data collection, formatted as an ISO TIMESTAMP, provides temporal context to each measurement.
- Node Identification Number (`node_id`): A unique identifier for each measurement node, formatted as an integer. This ID helps track data back to its source within the network's infrastructure.
- Network Identification Number (`network_id`): An integer indicating the operator for which the measurement is taken, thus distinguishing among various service providers.
- Service Identification Number (`service_id`): Specifies the technology utilized during the measurement, differentiating between 4G, 5G.
- Sent and Received Packets Count (`scnt`, `rcnt`): Counts of packets sent and received during the aggregation period, providing direct indicators of network packet loss and efficiency.
- Round Trip Time (`rtt`, `rtt_avg`): Measures the average time, in milliseconds, for a signal to travel to a destination and return, captured over 1-second and 5-minute aggregation intervals, respectively.

Metadata Collection

In addition to packet loss data, the study continuously collects metadata on network and device states to enrich the analysis.

- Signal Indicators: RSSI, RSRQ, and RSRP metrics provide detailed insights into the signal strength and quality, crucial for correlating network performance with signal characteristics.
- Network and Device Information: Includes SUBMODE, CID (Cell Identity), DEVICE_STATE, BAND, LTEFREQ, and EARFCN, offering comprehensive details on network operation and connectivity.
- Operational and Location Details: MODE, IPADDR, USBMODEM, LAC, and OPER data clarify the operational status of nodes and their geographical and network context.
- Advanced Metrics: IMSI, TX_POWER, and CELEVEL further deepen the dataset, capturing subscriber identity, transmission power, and coverage enhancement levels

The metadata is recorded through two distinct methodologies: Event-based data capture and Periodic recordings. Event data captures changes in metadata values at the precise moment an event occurs, typically marking a significant fluctuation or shift. Meanwhile, 1-minute bins consistently record each piece of metadata every minute, ensuring a comprehensive dataset over time. This dual approach allows for granular analysis of network conditions, facilitating the detection of anomalies.

4.1.2 Data Pre-processing:

Data Merging:

The preprocessing phase involves the approach to integrating data, which is essential for a comprehensive temporal analysis of network anomalies. This section explains the methodology used to combine Round Trip Time (RTT) data with event and 1-minute bin metadata.

RTT Data Aggregation: The Round-trip time (RTT) data is captured with millisecond precision and recorded every second. To create a unified dataset spanning the two months of study, a loop function extracts and aggregates the data. The compiled dataset provides comprehensive insights into network performance over time.

Metadata Integration: To integrate the RTT data with the metadata records, it is necessary to harmonize the temporal resolutions. To achieve this, two new timestamp columns are added to the aggregated RTT dataset. One column has a second granularity for aligning event metadata, while the other has a minute granularity for aligning 1-minute bin metadata. To adjust the timestamp column in the event metadata to second granularity, it is rounded down to the nearest second.

Data Merging:

- **Event Metadata Merging**: The aggregated RTT dataset is merged with the event metadata on three keys: the floored seconds timestamp, node ID, and network ID. This left-join operation ensures the retention of all RTT observations, aligning them with corresponding event metadata where available.
- **1-Minute Bin Metadata Merging:** Similarly, the 1-minute bin metadata is merged with the RTT dataset using the minute granularity timestamp, node ID, and network ID.

The rationale behind utilizing left joins is to maintain the integrity of the RTT dataset, ensuring no observation is omitted. Columns introduced from the metadata that do not find a corresponding match in the RTT dataset are filled with NaNs, to be addressed in subsequent preprocessing steps.

Data Cleaning

The data preparation involves several steps to clean and organize the dataset for analysis. Missing values are managed using a forward fill technique where subsequent NaNs are replaced with the closest prior non-NaN value, except for the initial sequence of NaNs which are handled by referencing the 1-minute bin records. Event metadata timestamps are aligned by flooring to the nearest second and adjusting any additional events within the same second to the following second to maintain temporal uniqueness. RTT data is validated to ensure both sent and received packet counts are non-zero, discarding any invalid record. Data reflecting normal network device operations is isolated by filtering for an operational mode value of 6, and any extraneous Bin metadata columns are removed to streamline the dataset. Finally, the dataset is sorted chronologically to facilitate time-series analysis, ensuring the data is primed for detailed network performance and anomaly investigation. The algorithm below shows the step-by-step process for data cleaning

4.2 Model Development

Machine learning (ML) models are essential for anomaly detection in mobile networks. They use large data volumes in networks to improve their reliability, efficiency, and security. This section outlines the criteria for selecting suitable models and provides details about the chosen models for development.

Theorectical foundation

The conceptual framework for the model presented is a synthesis of Convolutional Neural Networks and Long Short-Term Memory networks. This dual-structure approach seeks to capitalize on the respective strengths of each: CNNs are employed to identify patterns over short-term sequences, while LSTMs are engaged to get the dependencies across extended temporal intervals. The rationale

Algorithm 1 Comprehensive Data Processing for Network Anomaly Detection

Require: Nornet Edge (NNE) platform data, measurement period
Ensure: Cleaned and processed dataset ready for analysis

Initialize empty dataset D {Data Collection}
for each day in measurement period **do**
  $D_{day} \leftarrow$ Load RTT and packet loss data from CSV
  $D \leftarrow D \cup D_{day}$ {Aggregate daily data}
end for{Data Combination}
Prepare D with second and minute granularity timestamp columns
for each type of metadata in [event, 1min bin] **do**
  for each metadata file M corresponding to the current type **do**
    if type is event **then**
      Adjust M timestamps for granularity and floor to seconds from milli-seconds and resolve duplicates
    end if
    Merge D with M based on timestamp, node ID, and network ID
  end for
end for{Data Cleaning}
for each record in D **do**
  if sent or received packet count is 0 OR operational mode $\neq 6$ **then**
    Discard record from D
  end if
end for
Drop all Bin metadata columns no longer needed from D
Sort D by timestamp
return D

for this architecture is driven by the capabilities of CNNs and LSTMs as detailed in the background section is further illustrated by examples of their applications in the literature review.

4.2.1 Centralised Architecture

The centralized implementation phase of the research is crucial in determining the most suitable model architecture and hyperparameter setup for anomaly detection within mobile networks. This stage involves rigorous experimentation with multiple machine learning models and their corresponding hyperparameters to select our base model.

It also provides an opportunity to contrast the performance of the centralized model with that of the federated learning model.

Such comparisons are crucial in evaluating the validity and potential benefits of federated learning approaches. Findings from this phase are instrumental in guiding the selection and fine-tuning of models, ultimately enhancing the assessment of federated learning as a viable solution for the unique demands of mobile network anomaly detection.

Data Preparation

For data preparation, a two-week dataset from three nodes is collected and merged. The data is sorted based on the timestamp column. Additional columns are created to assist the model further. Rolling mean, median, and standard deviation of the Round Trip Time (RTT). Day of the week and weekend indicators are added. As per findings from Nornet research, any Round-Trip Time (RTT) values exceeding 45 milliseconds are flagged as anomalies, while those below this threshold are considered normal observations. Subsequently, a new column named anomalies is created, where anomalies are labeled as 1 and non-anomalies as 0.

The data is then split chronologically into training (60%), validation (20%), and test (20%) sets, considering the class imbalance. To address normalization, the labels are separated from the data, and a standard scaler is applied, fitted on the training set, and transformed on both validation and test sets to prevent data leakage.

Next, the data is converted into sequences using the Keras time series dataset function, with a timestep of 100 and a batch size of 128, incorporating labels to create batch datasets for training, validation, and testing.

Platform

Given the dataset's granularity and the training's computational demands, the implementation is carried out on Google Colab. This platform provides the necessary computational resources to

handle the intensive training processes efficiently.

Experiments are primarily conducted using 50GB of CPU and Google T4 GPU for training the model. This setup ensures the feasibility of training complex models and conducting extensive experimentation to optimize model performance for this study.

Model Selection

Faced with the challenges of high dimensionality and class imbalance in a labeled dataset, various approaches ware tested. Unsupervised methods are initially used but failed to distinguish between normal and anomalous data effectively. Transitioning to supervised learning, CNN alone is tried but does not perform adequately. The introduction of an LSTM layer to the model improved its ability to recognize temporal dependencies and anomalies. This combination of CNNs and LSTMs ultimately enhances the model's anomaly detection capabilities.

Base Model

The architecture of the developed model features a sequential model configuration, carefully structured with a series of layers to extract and learn patterns from mobile network data intricately. The configuration is as seen below.

Layer Descriptions

The model architecture begins with an input layer that defines the shape of the data for the network. It then utilizes three convolutional layers paired with max-pooling and batch normalization to extract spatial features. To capture temporal relationships, an LSTM layer is integrated into the model. Dropout layers are employed throughout to reduce the risk of overfitting. The architecture culminates in an output layer with a single neuron using a sigmoid activation function for binary anomaly detection. The ReLU activation function is used in all except the output layer to introduce non-linearity. All the parameters for each layer can be seen in Figure 4.3

Binary Focal Cross Entropy Loss is utilized to address class imbalances in binary classification tasks by modifying traditional Binary Cross Entropy Loss with a focusing parameter α. This parameter allows the model to focus more on difficult, misclassified samples rather than the majority class, thereby minimizing bias and improving model accuracy on imbalanced datasets (T.-Y. Lin et al., 2017).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 100, 32)           1280

 max_pooling1d (MaxPooling1  (None, 50, 32)            0
 D)

 batch_normalization (Batch  (None, 50, 32)            128
 Normalization)

 dropout (Dropout)           (None, 50, 32)            0

 conv1d_1 (Conv1D)           (None, 50, 64)            6208

 max_pooling1d_1 (MaxPoolin  (None, 25, 64)            0
 g1D)

 batch_normalization_1 (Bat  (None, 25, 64)            256
 chNormalization)

 dropout_1 (Dropout)         (None, 25, 64)            0

 conv1d_2 (Conv1D)           (None, 25, 128)           24704

 max_pooling1d_2 (MaxPoolin  (None, 12, 128)           0
 g1D)

 batch_normalization_2 (Bat  (None, 12, 128)           512
 chNormalization)

 dropout_2 (Dropout)         (None, 12, 128)           0

 lstm (LSTM)                 (None, 128)               131584

 dropout_3 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 164801 (643.75 KB)
Trainable params: 164353 (642.00 KB)
Non-trainable params: 448 (1.75 KB)
_________________________________________________________________
```


Figure 4.3: Model Summary

Model Training and Evaluation

The trained model undergoes rigorous evaluation to assess its performance.

Training:

Table 4.1: Summary of Model Training Phases

Phase	Number of Epochs	Optimizer and Learning Rate	Loss Function (Alpha, Gamma)
Initial Training	15 epochs	Adam, Learning Rate = 0.00001	Binary Focal Loss (0.8, 2)
Additional Training	20 epochs	Adam, Learning Rate = 0.00001	Binary Focal Loss (0.8, 2)

Evaluation Metrics:

1. **Recall:** Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive cases that the model correctly identifies. It is calculated using the formula:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

2. **Precision:** Precision is a measure that evaluates the model's ability to correctly identify positive instances. It is computed as the ratio of true positives to the sum of true positives and false positives, reflecting the model's proficiency in minimizing false positive predictions. The precision is computed as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

3. **Accuracy:** Accuracy represents the overall correctness of the model's predictions across all classes. It measures the proportion of correctly classified instances to the total number of instances evaluated. The accuracy is given by:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}}$$

4. **F1 Score:** The F1 Score is the harmonic mean of precision and recall, giving a balance between both.

$$\text{F1 Score} = 2 \times \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \right)$$

These evaluation metrics provide insights into different aspects of the model's performance, helping to gauge its effectiveness in detecting anomalies (Naidu et al., 2023).

4.2.2 Distributed Framework Implementation

For implementing federated learning, various frameworks are available, among which we chose Flower. Flower stands out for its ease of configuration and integration into existing machine-learning workflows. Flower provides a distributed system for federated learning, allowing seamless coordination between server and client components. Through Flower, model training tasks are efficiently distributed across edge devices, enabling collaborative learning while ensuring data privacy and security.

Further functionalities and integration details are explained in Beutel et al., 2020 which describes the process of using Flower for federated learning implementations.

Platform

In our federated learning setup, we utilize four virtual machines within the same network. Each virtual machine is provisioned with 35 GB of RAM and runs Ubuntu 22.04 as the operating system. This infrastructure provides the necessary computational resources to handle the distributed model training tasks across multiple devices while ensuring smooth coordination and communication between the server and client components.

However, it's important to note that the available resources are limited, which constrains our ability to perform training on the entire dataset in a single iteration. This necessitates careful optimization and partitioning of the data to ensure efficient training within the resource constraints.

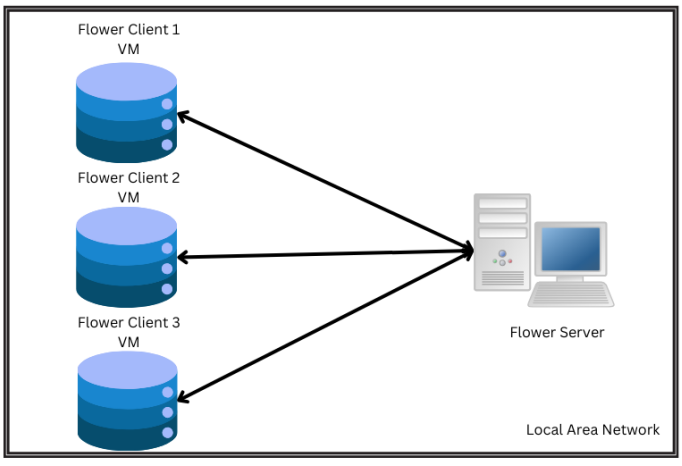


Figure 4.4: Flower Implementation Design

Please visit the following link to access the source files and detailed implementation:

GitHub Repository: Thesis-ANO-FED-CNN-LSTM

Model Training and Evaluation

In the federated implementation, each client's Python file comprises data preprocessing, model definition, and the fitting function, all tailored to its local dataset. The parameters and loss function mirror those described in the base model. Binary Focal cross-entropy serves as the loss function, prioritizing the positive class for most nodes during training

Table 4.2: Training Parameters for Nodes

Parameter	**Bud**	**Tromso**	**Gelio**
Epochs	30	30	30
Early Stopping	Yes	Yes	Yes
Patience	6	6	6
Restore Best Weights	Yes	Yes	Yes
Learning Rate	0.0001	0.0001	0.0001
Loss Function	Binary Focal Cross-Entropy	Binary Focal Cross-Entropy	Binary Focal Cross-Entropy
Alpha	0.25	0.8	0.8
Gamma	2	2	2

During training, each client virtual machine trains its model on its local dataset. The trained model weights are then transmitted to the central server. Utilizing federated averaging (FedAvg), the server aggregates these weights to generate the global model. Subsequently, the global model weights are distributed back to each client, where they are employed to evaluate the model on their respective test datasets. This iterative process, termed rounds, continues.

For evaluation, we adhere to the same set of metrics: F1 Score, precision, recall, and accuracy. Following each round, these metrics are computed for each client on their respective test data. A weighted average is then calculated, aggregating these metrics across all clients and providing a comprehensive view of model performance.

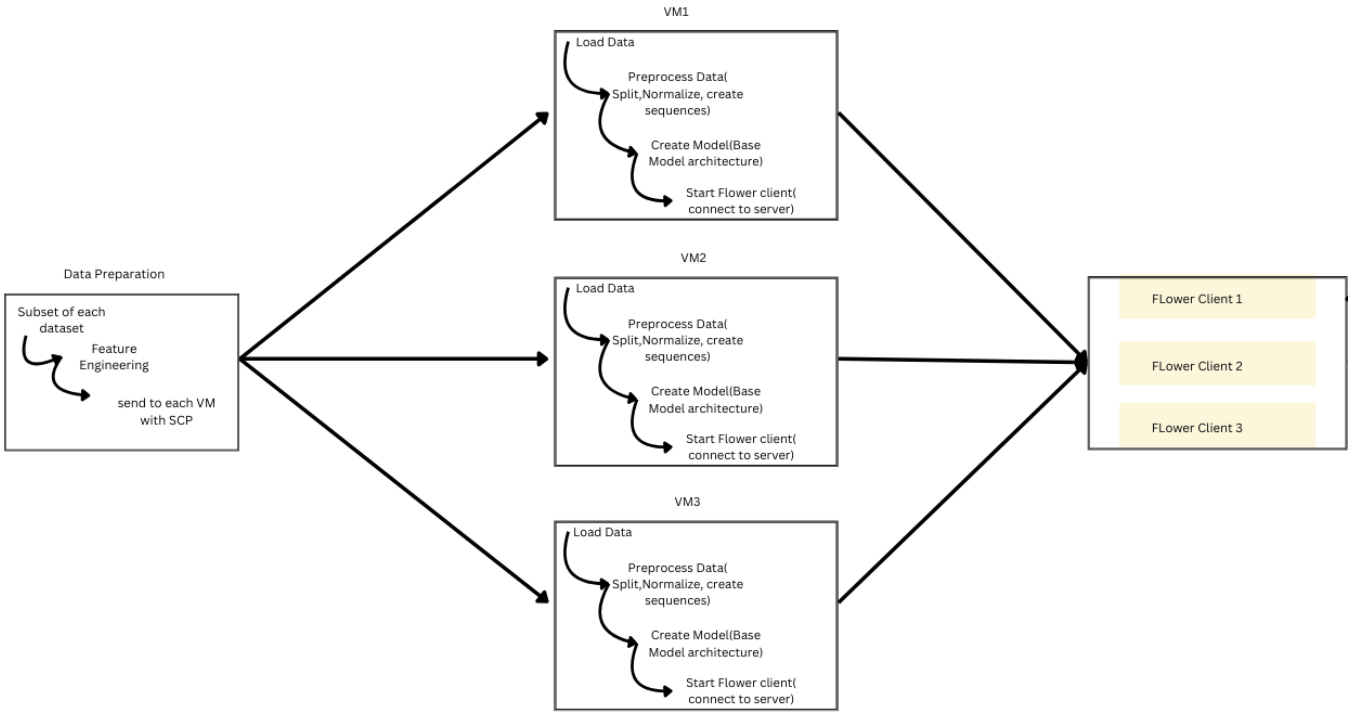


Figure 4.5: The process of creating a federated client for each node

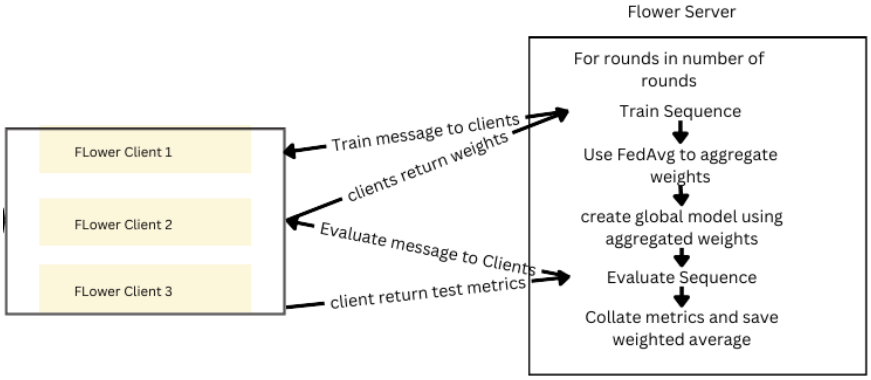


Figure 4.6: Federated Training process between clients and server

Chapter 5

Results

5.1 Data Exploration

Descriptive statistics

The boxplot depicted in Figure 5.1 illustrates the distribution of Round-Trip Time (RTT) across various network nodes for a selected week, highlighting differences in network performance among different locations. Notably, the node in Tromsø (4135) exhibits the highest median RTT of approximately 50.8 milliseconds, suggesting slower network response times which might be indicative of less efficient infrastructure or greater network congestion compared to other nodes. In contrast, nodes in Oslo (4144 and 4147) show significantly lower median RTTs, around 31.6 and 32.1 milliseconds respectively, indicating more robust network conditions. The node in Bud (4152) and the node in Gelio (4137) also show median RTTs of 51.1 and 37.4 milliseconds, respectively, suggesting moderate performance relative to the others.

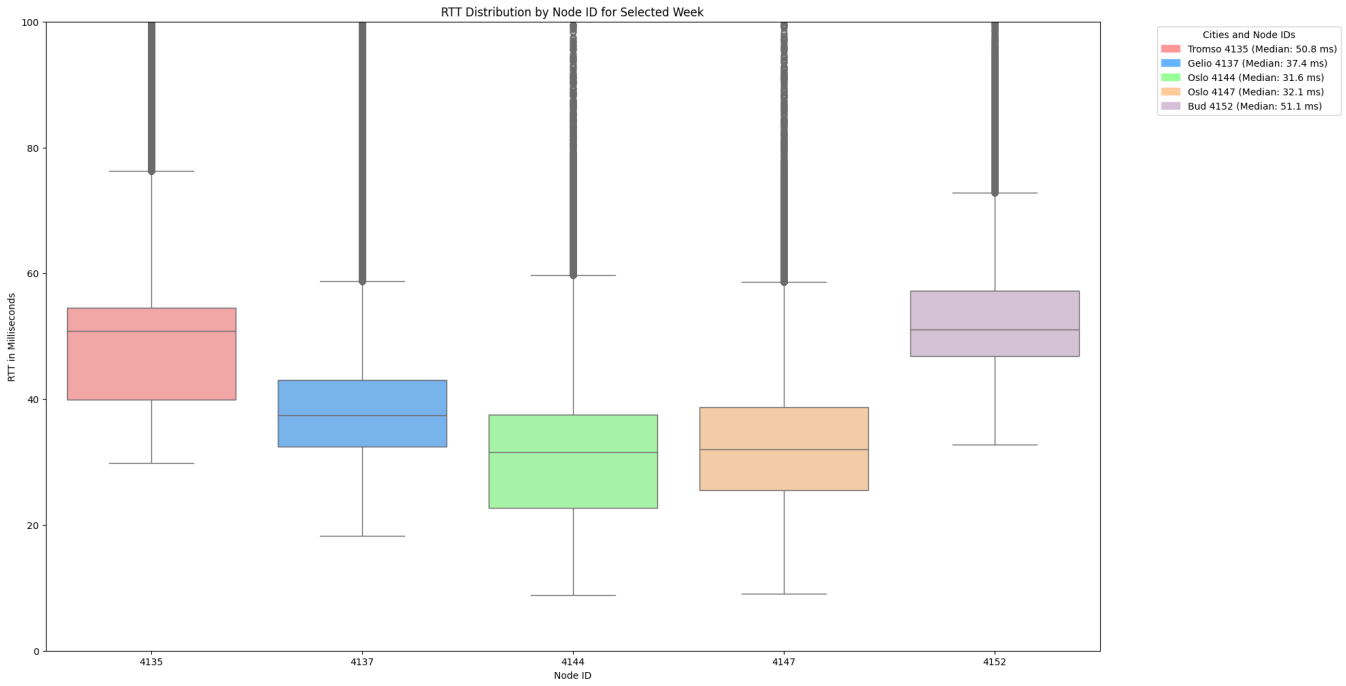


Figure 5.1: RTT Distribution by Node ID.

Network Performance Analysis by Time and Day

Key Observations:

Node 4152 Figure 5.2: RTT peaks on Sundays between 10 AM and 12 PM, reaching up to 85 milliseconds, whereas Mondays show slightly later peaks with similar maximum RTT, indicating a routine shift at the start of the workweek.

Node 4137 Figure 5.2: Displays higher RTT on Sunday mornings (up to 80 milliseconds) compared to Mondays (maximum of 70 milliseconds) during similar times, suggesting different usage patterns on weekends versus weekdays.

Daily Trends: Weekdays, especially Mondays, see a consistent rise in RTT during business hours (9 AM to 5 PM), while weekends exhibit RTT spikes potentially due to leisure-related high-bandwidth activities.

The observed temporal patterns suggest both daily and weekly RTT seasonality that might be associated with user behavior and network load. Peaks in latency often coincide with hours that could correspond with increased network usage.

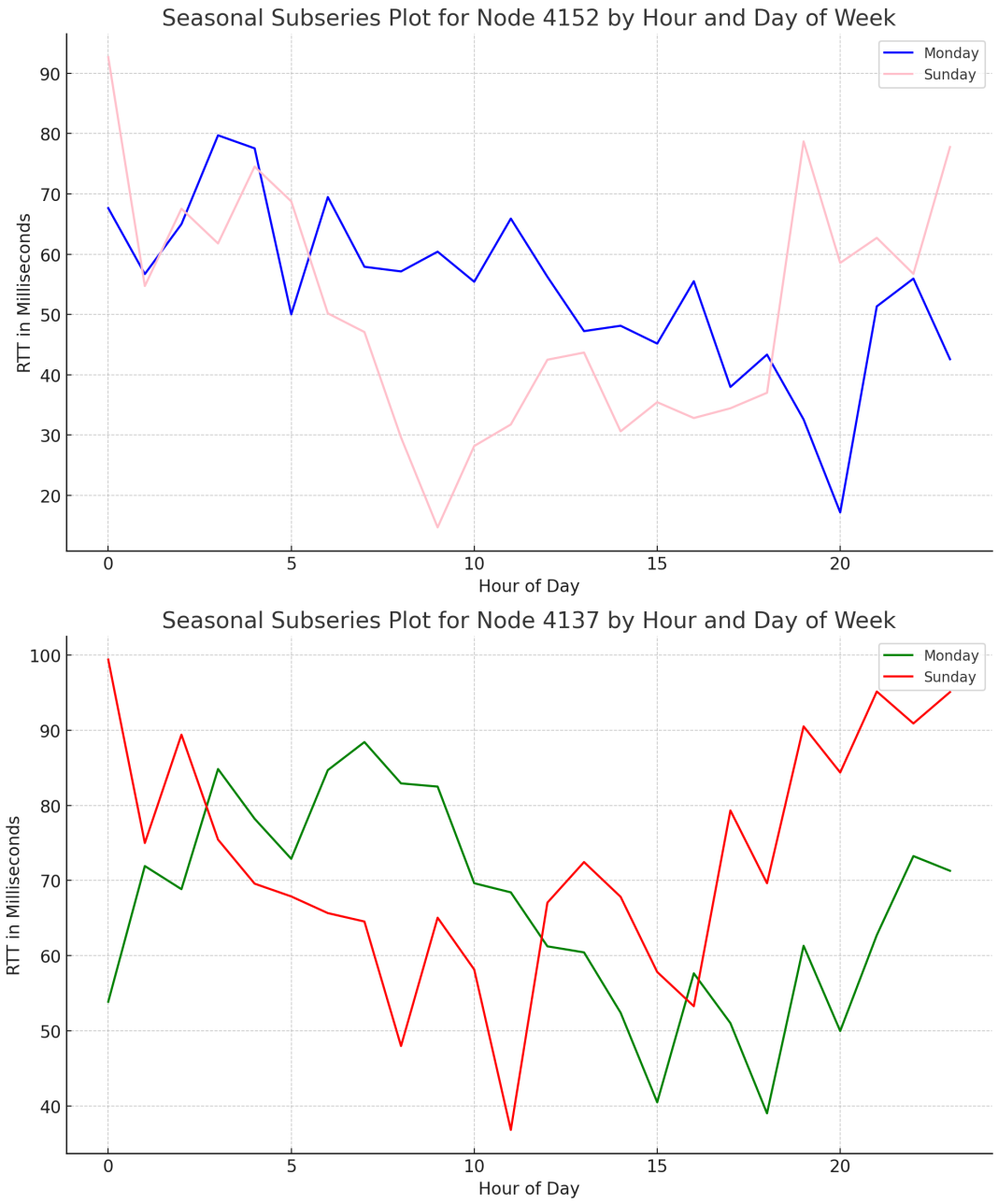


Figure 5.2: Seasonal Subseries Plot for Node Bud 4152 and Gelio 4137 illustrating RTT across different hours of the day and days of the week.

RTT Distribution

The RTT distribution plot illustrates the spread of RTT values across all nodes. Approximately 20% of nodes exhibit an RTT equal to or below 15 milliseconds, indicating a significant fraction with low RTT. Between 15 and 35 milliseconds, there's a steep increase in RTT values, with the majority falling within this range. Beyond 35 milliseconds, the curve levels off, suggesting fewer nodes with higher RTT values. By the 50-millisecond mark, around 80% of nodes have an RTT

below this level, and nearly all nodes (close to 100%) have an RTT below 70 milliseconds. Overall, most nodes display moderate RTT, but there's a tail distribution indicating a smaller proportion experiencing higher RTT values.

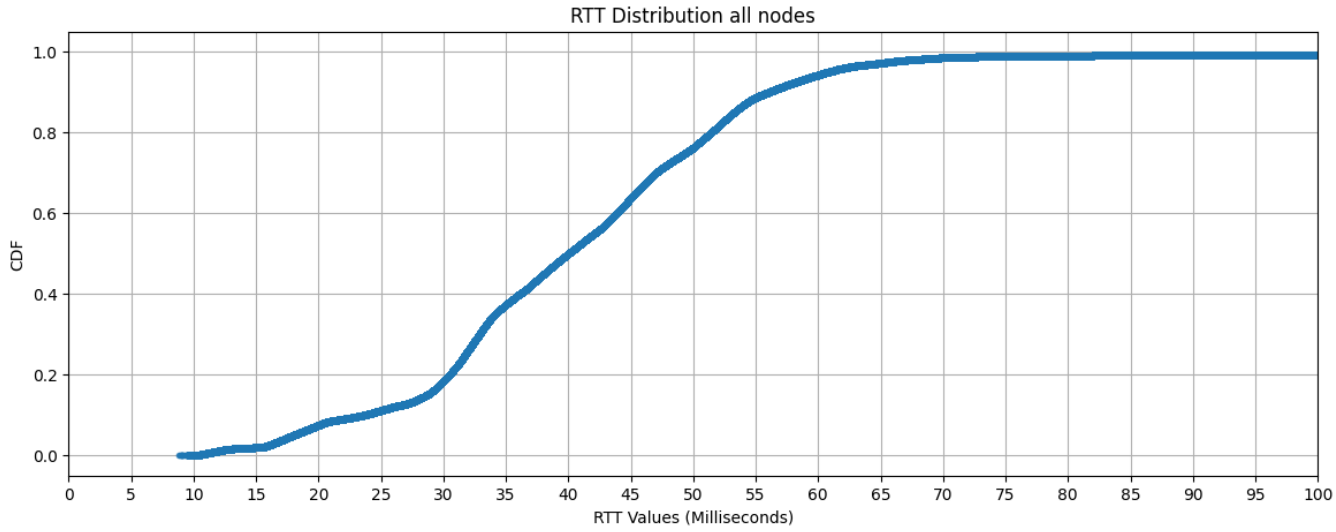


Figure 5.3: RTT distribution plot for all the nodes considered in this study.

5.2 Results For Centralised Architecture

We combined 2 weeks of data from the nodes to use for the centralized training

Anomaly Label Distribution

The distribution of anomaly labels within our dataset shows abnormal behavior detected by our anomaly detection model. Figure 5.4 outlines the breakdown of anomaly labels into two categories: negative anomalies, denoting instances classified as normal behavior, and positive anomalies, indicating deviations from the norm.

Our dataset comprises a total of 3,606,412 labeled instances, with a significant majority of 2,502,441 instances classified as negative anomalies.

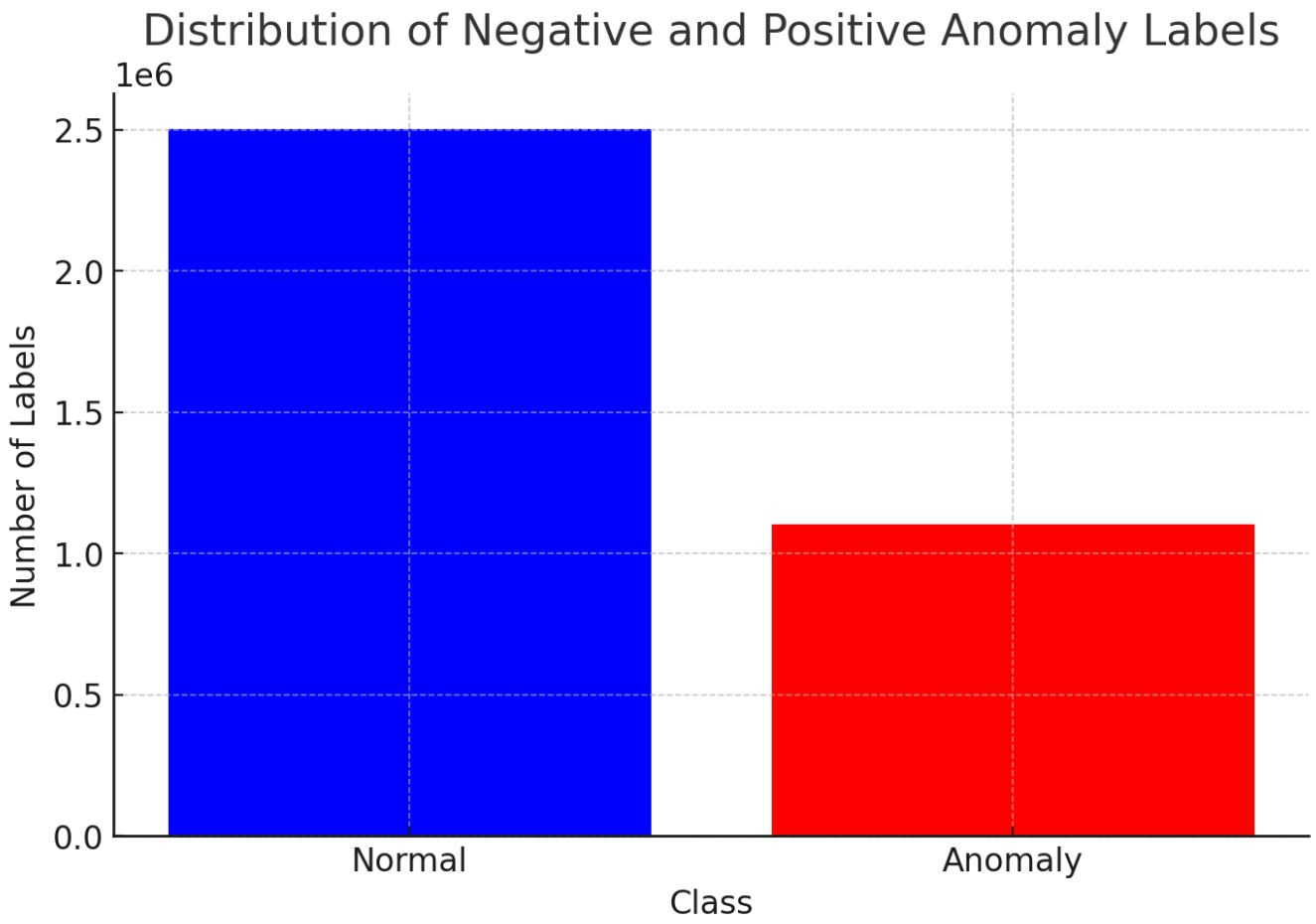


Figure 5.4: Distribution showing class imbalance in the dataset.

Model Training

In the initial training phase, the model is trained for a total of 15 epochs, with early stopping implemented as a precaution against overfitting. Early stopping is prompted by an observed convergence in the validation and training losses, indicating that further training does not result in significant improvements in the validation loss (Figure 5.5).

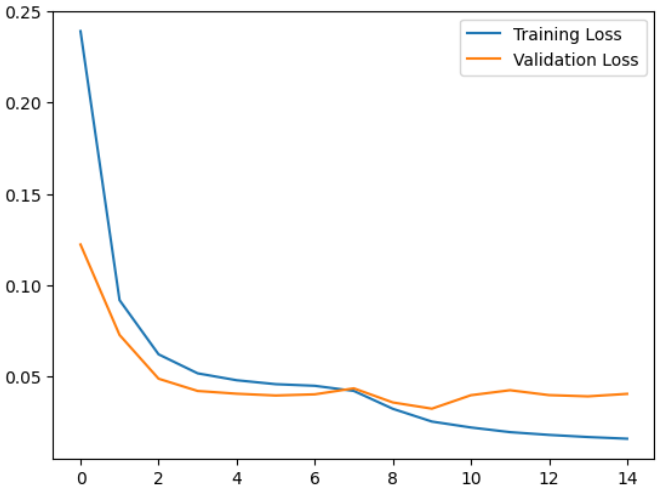


Figure 5.5: The model's training and validation loss over the initial 15 epochs.

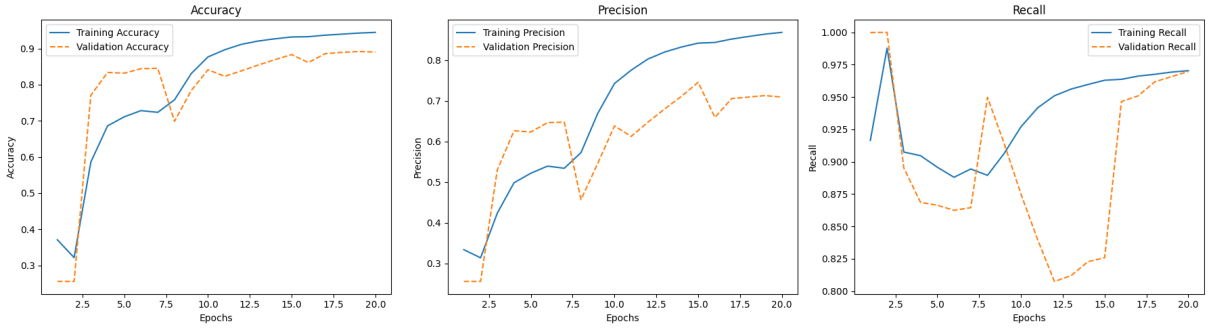


Figure 5.6: The training and validation results over the initial 15 epochs.

During the initial epochs, the validation loss is lower than the training loss, which is uncommon but can occur due to the effects of regularization or differences in the distribution of the training and validation datasets. After the early stopping criterion is met, further training of 20 epochs is conducted. This extended training leads to a continued decrease in the validation loss, indicating an improvement in the model's generalization ability (Figure 5.7).

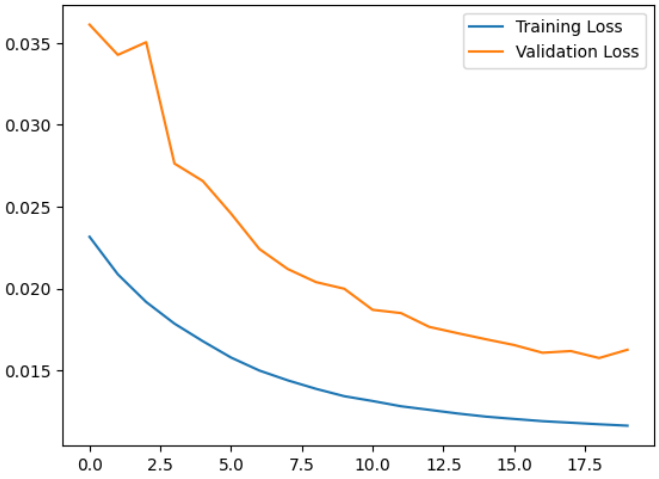


Figure 5.7: Continuation of the model's training for an additional 20 epochs after early stopping.

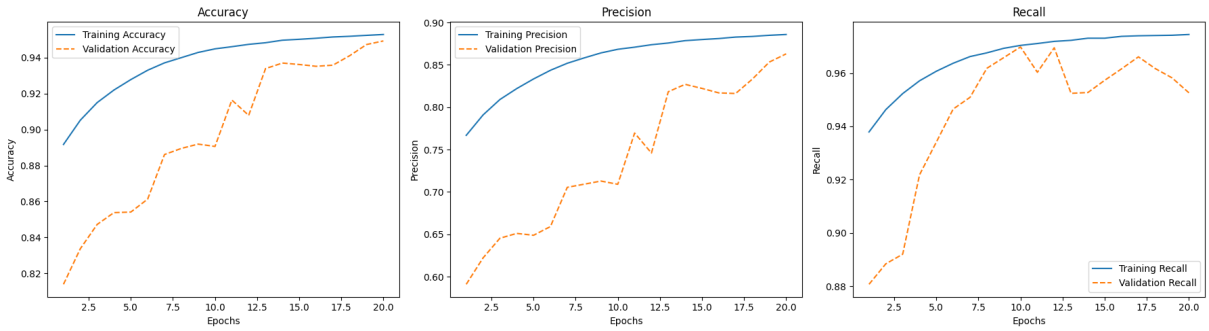


Figure 5.8: Metrics for additional 20 epochs after early stopping.

The decrease in validation loss over the additional epochs validates the decision to extend training beyond the early stopping point. The final epochs indicate the validation loss approaching a plateau, suggesting the model's optimal performance given the current architecture and dataset.

The final evaluation of the centralized learning approach yielded the following metrics, as summarized in Table 5.1:

Table 5.1: Metrics for centralized training

Label	Accuracy	Precision	Recall	F1 Score
Centralized Training	0.877	0.904	0.714	0.798

The centralized training approach achieved an accuracy of 87.7%, indicating the proportion of correctly classified instances out of the total dataset. Precision, which measures the proportion of true positive predictions among all positive predictions, is calculated at 90.4%. Recall, representing

the proportion of true positive predictions among all actual positive instances, stood at 71.4%. The F1 score, a harmonic mean of precision and recall, is determined to be 79.8%.

These metrics provide a comprehensive assessment of the model's performance in the centralized learning setting, serving as a baseline for comparison with other approaches.

5.3 Results For Distributed Architecture Via Federated Learning

We present the initial findings from the application of federated learning to our anomaly detection model. The primary goal is to leverage decentralized data across different nodes to enhance the model's robustness and generalizability without compromising privacy and data security.

5.3.1 Training Observations

We start the training with 20 rounds, with early stopping integrated to mitigate the risk of overfitting. For clarity, we focus on the first round of training across three different nodes, Gelio 4137, Tromsø 4135, and Bud 4152. The plots provided below represent the training and validation loss over the epochs for this initial round.

Node Bud 4152: The training history for Node Bud 4152 shows a steady decrease in training loss along with a relatively flat validation loss after an initial adjustment (Figure 5.9).

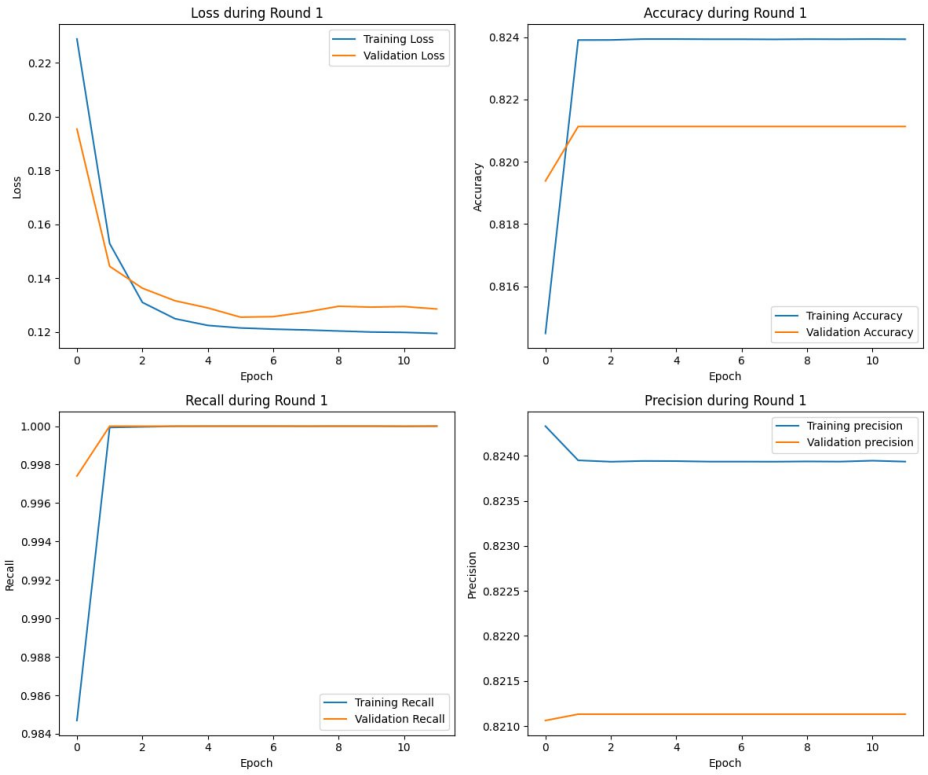


Figure 5.9: Bud node 4152 training and validation.

Node Tromsø 4135: Node Tromsø 4135 displays a more consistent reduction in both training and validation loss. This consistency points to a well-generalizing model on this node's data (Figure 5.10).

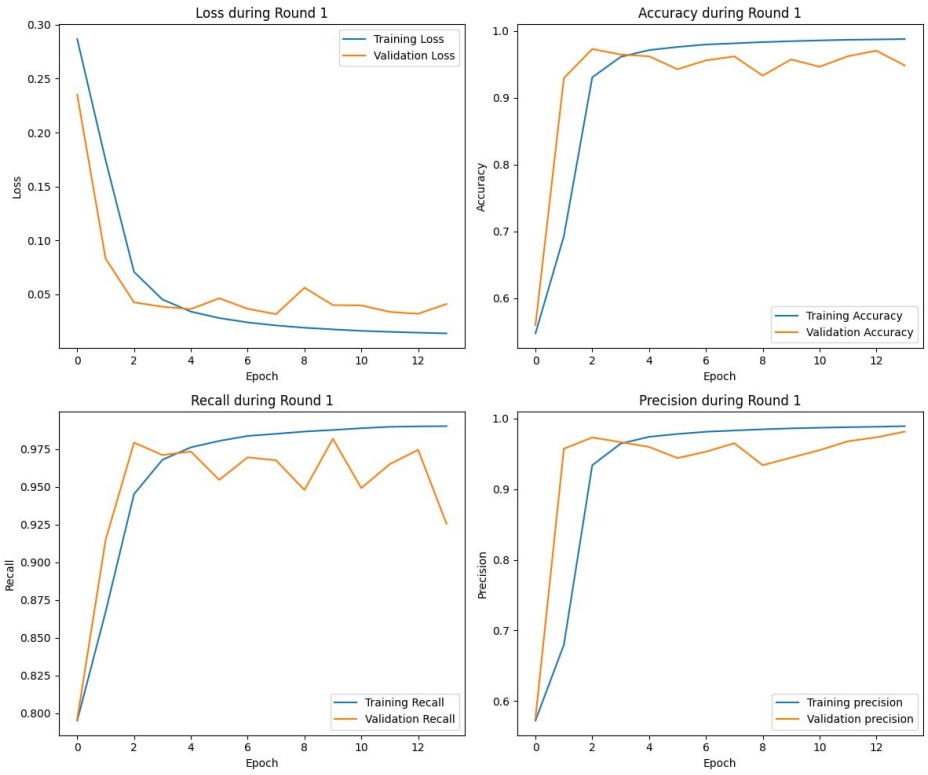


Figure 5.10: Tromso node 4135 training and validation.

Node Gelio 4137: For Node Gelio 4137, the training history illustrates an initial steep decline in training loss, which is indicative of rapid learning. However, the corresponding validation loss demonstrates higher variability. (Figure 5.11).

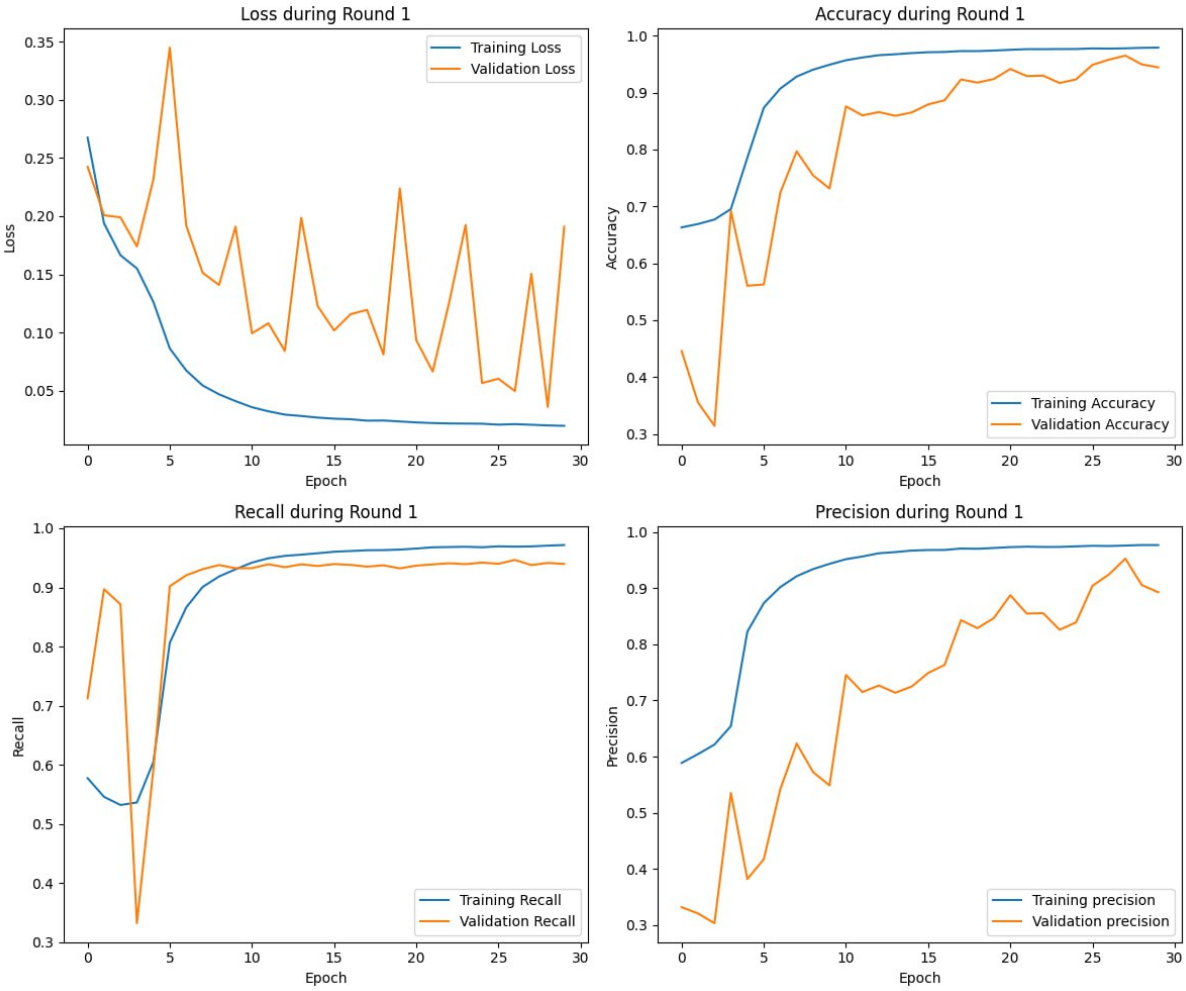


Figure 5.11: Gelio node 4137 training and validation.

5.3.2 Evaluation Results

The evaluation of our federated learning model's performance over 20 rounds during training is done through the progression of binary accuracy, precision, and recall.

Local Evaluation

Figure 5.12 illustrates the performance of the global model on each node after every round. A consistent and stable pattern emerges across all nodes for recall, reflecting our deliberate focus on ensuring high recall rates before prioritizing improvements in other parameters in each successive round.

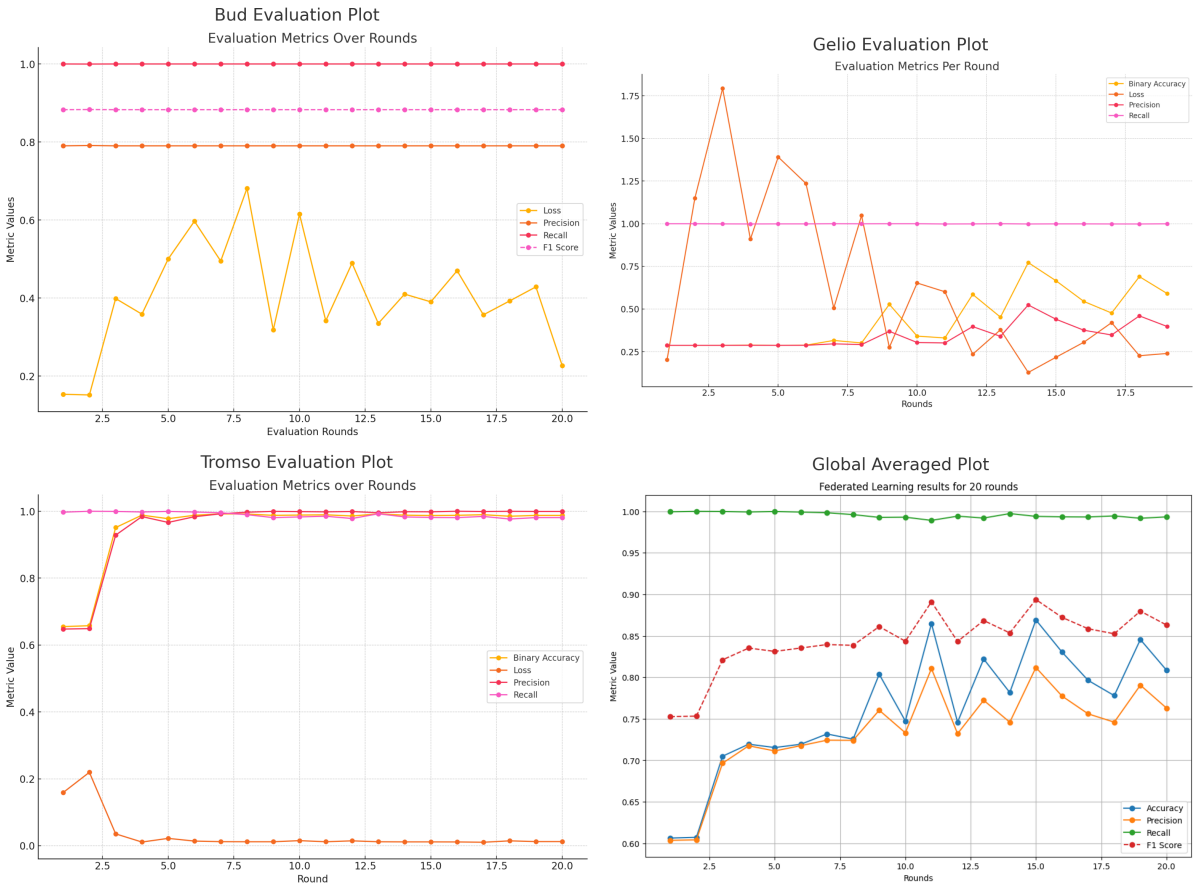


Figure 5.12: The local evaluation of the global model at every round and the global averaged result.

Global Averaged result

Figure 5.12 presents the averaged global test metrics plot, showcasing the fluctuations and trends observed in binary accuracy, precision, and recall across the training rounds.

Table 5.2 provides a detailed breakdown of the metrics for each round, including binary accuracy, precision, recall, and F1 score.

Table 5.2: Federated Learning Results

Round	Accuracy	Precision	Recall	F1 Score
1	0.6062	0.6035	0.9994	0.7523
2	0.6072	0.6041	0.9999	0.7530
3	0.7049	0.6964	0.9998	0.8211
4	0.7194	0.7174	0.9992	0.8350
5	0.7153	0.7112	0.9997	0.8307
6	0.7193	0.7176	0.9991	0.8352
7	0.7317	0.7242	0.9982	0.8403
8	0.7254	0.7240	0.9960	0.8385
9	0.8037	0.7603	0.9926	0.8605
10	0.7473	0.7331	0.9929	0.8433
11	0.8647	0.8105	0.9890	0.8917
12	0.7459	0.7320	0.9941	0.8436
13	0.8220	0.7723	0.9918	0.8662
14	0.7816	0.7458	0.9972	0.8539
15	0.8691	0.8118	0.9939	0.8928
16	0.8303	0.7774	0.9933	0.8714
17	0.7962	0.7559	0.9931	0.8567
18	0.7779	0.7459	0.9944	0.8517
19	0.8455	0.7905	0.9916	0.8790
20	0.8085	0.7629	0.9932	0.8608

Chapter 6

Discussion

In this thesis, federated learning is applied to the task of anomaly detection in mobile networks to assess its effectiveness in anomaly detection in high-dimensional data across distributed nodes. The research reveals significant variations in latency and signal quality metrics (RTT, RSSI, RSRQ, and RSRP) across diverse geographic locations, such as Bud, Gelio, Tromsø, and Oslo. These findings are important for understanding how geographical and infrastructural factors influence network performance.

This analysis evaluates the performance of centralized and federated learning models based on the metrics, namely, precision, recall, and F1 score. While federated learning demonstrates the ability to enhance data privacy and support localized data processing, its performance varies across different nodes.

These observations suggest that federated learning could be viable for real-world applications; however, they also show the complexity of deploying such technologies effectively across geographically dispersed data. Moving forward, this discussion places these results within the broader context of mobile network management and anomaly detection, examining the implications of these findings for deploying machine learning techniques to enhance network reliability and performance. It addresses the challenges and limitations encountered during the study.

6.1 Interpretation of Key Findings

6.1.1 RTT Variations

Geographical Impact

The Round-Trip Time (RTT) metrics observed for different nodes across Norway provide insights into the relationship between geography, infrastructure, and network performance. RTT variations

across nodes in cities like Bud, Gelio, Tromsø, and Oslo likely reflect a combination of geographical and infrastructural differences. The mountainous and fjord-rich landscapes of Northern Norway, where Tromsø and Bud are located, may present natural barriers to signal propagation, contributing to higher RTT values.

Adding to the geographical influence is the possible impact of server proximity. With the central data servers likely situated closer to Oslo, the data packets have a shorter distance to travel, resulting in lower RTT values for the Oslo node. This is less so for nodes in the north, such as Bud and Tromsø, where the greater physical distance from the servers naturally extends transmission times. Gelio's more evenly distributed RTT values, bridging the lower threshold and the defined anomaly range, indicate an intermediate position in terms of server proximity and the consequent network latency.

Infrastructure quality further compounds these variations. The presence of more modern or well-maintained network equipment in Oslo, as opposed to potentially older infrastructure in the north, can also contribute to the differing RTT profiles.

These factors show the importance of considering both natural geography and the strategic placement of network infrastructure in ensuring optimal network performance across diverse regions.

Refer to Figure 6.1for a visual representation of the RTT distributions:

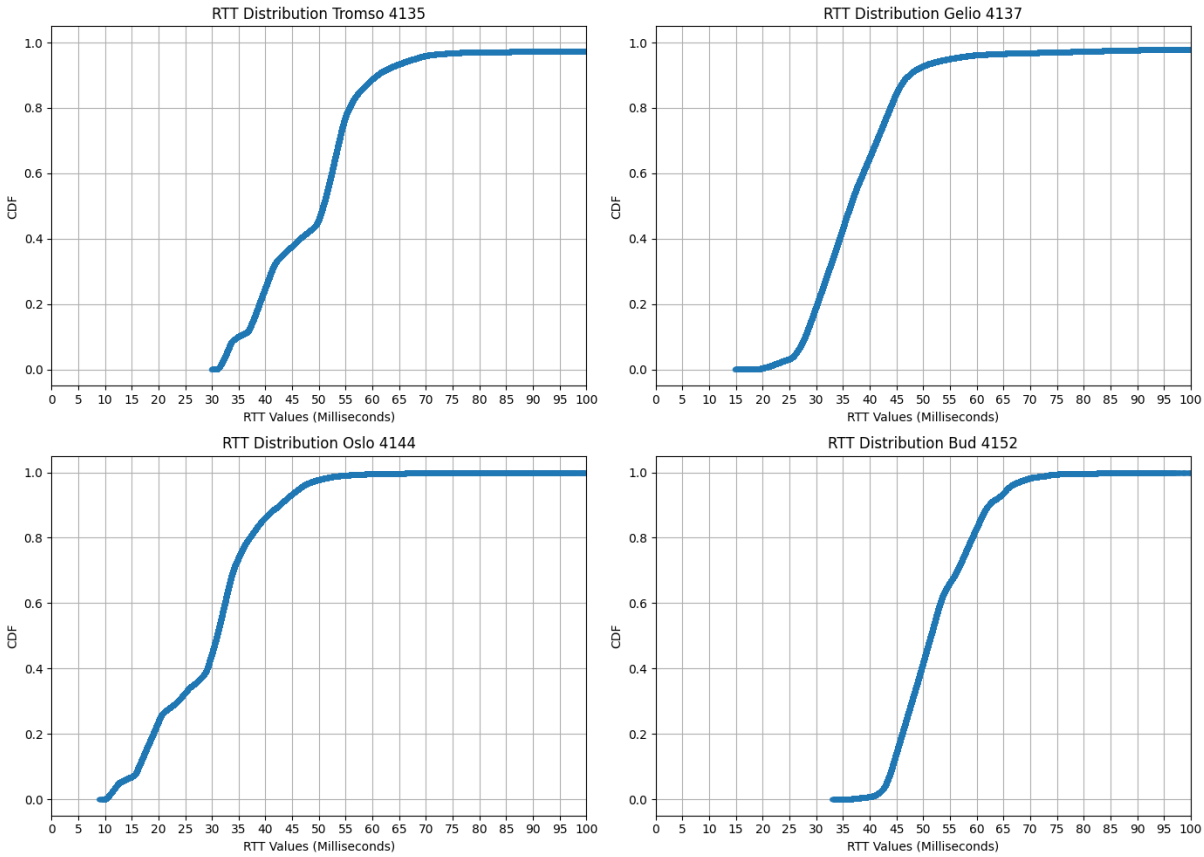


Figure 6.1: RTT distribution for each node.

Network Load and User behavioral pattern

The results as seen in Figure 5.2 show clear temporal variations in RTT across different nodes, which not only reflect network performance fluctuations but also align closely with user behavioral patterns across different days of the week.

Temporal Variations in RTT

Patterns Observed: The peak RTT times on Sundays compared to Mondays suggest a shift in the type of network usage. On Sundays, high RTT peaks likely correspond to leisure activities that require high bandwidth, such as streaming videos or online gaming. This contrasts with Monday's peak times, which may be influenced by professional or educational activities as the workweek begins.

Implications: These observations imply that network infrastructure is subjected to varying types of stress depending on the day. Service providers might need to consider adaptive bandwidth allocation strategies that cater differently to weekdays versus weekends.

Correlation with User Behavior:

Behavioral Insights: The direct correlation between RTT peaks and specific times of day supports the notion that network performance issues are not merely technical but are significantly influenced by user behavior.

Strategic Adjustments: Network operators could benefit from these insights by scheduling maintenance during low-usage periods and potentially offering differentiated service plans that optimize network performance based on expected usage patterns.

6.2 Data Dimensionality

The dimensionality of the data used in this study is described in Table 6.1. The size of the datasets underlines the challenge of high-dimensional data management and the implications for computational resources.

Table 6.1: Dataset Dimensions and File Sizes

Filename	Rows	Columns	Size (bytes)	Size (MB)
bud_feature_eng.csv	4,969,098	16	525,791,286	501.43
oslo4144_feature_eng.csv	5,226,690	16	402,593,547	383.94
Tromso_feature_eng.csv	5,216,801	16	403,708,883	385.01
gelio_feature_eng.csv	5,221,451	16	402,142,735	383.51

The table shows that the datasets consist of millions of observations across numerous variables for each node. Centralizing such extensive data for model training would entail significant computational resources and potentially create bottlenecks due to data transfer loads. Moreover, centralizing data raises concerns about user privacy and data security.

Federated learning offers a viable solution to these challenges. By distributing the training process across multiple nodes, computational resources are utilized to learn locally from the data. This approach significantly reduces the requirement to transfer large volumes of data, as only model updates are communicated back to a central server. Consequently, this not only conserved bandwidth but also mitigated the risks associated with data privacy breaches.

The implementation of federated learning, thus, allows for computational and resource efficiency, with the added benefit of maintaining the privacy of the underlying data. This distributed approach to machine learning proves to be particularly advantageous when dealing with high-dimensional datasets as demonstrated in this study.

6.2.1 Centralised Architecture

The centralized learning model demonstrates robust performance, with key metrics outlined in Table 5.1. The model achieves an accuracy of 87.7%, indicating a high level of correct anomaly detections,

essential for network operation reliability. Precision is particularly high at 90.4%, ensuring that the model effectively identifies actual anomalies, minimizing false positives.

However, the recall of 71.4% suggests that the model could miss some true anomalies, indicating a potential area for improvement. The F1 score of 79.8% reflects a balance between precision and recall, highlighting the model's effectiveness in handling anomaly detection tasks.

We use these results as a benchmark for evaluating our federated learning model.

6.2.2 Distributed Architecture

The results from the federated learning approach are detailed in Table 5.2. Over 20 rounds of training, we observe a notable improvement in all metrics. Initially, the accuracy starts at 60.62% but sees an increase, peaking at 86.91% in the 15th round with a precision of 81.18% before stabilizing around 80.85% by the 20th round.

The recall consistently being high indicates the model's effectiveness in identifying anomalies. The highest recorded F1 score is 89.17%, reflecting a balanced model that is both precise and sensitive to detecting true positives.

For each training round, the model parameters were saved, allowing for the examination of specific iterations and the potential reuse of the most effective configurations. This systematic parameter saving helps assess the model's evolution and convergence over time.

6.2.3 Comparative Analysis with Centralized Learning

This study compares federated and centralized learning models to establish a baseline for performance evaluation. The centralized model metrics are presented in Table 5.1, which serve as a benchmark against the federated model results detailed in Table 5.2.

While the federated learning model exhibits a slightly lower accuracy than the centralized approach, it achieved comparably high precision and recall values. Most notably, the federated model consistently records higher recall rates throughout the training rounds. This high recall indicates the model's robustness in anomaly detection across distributed datasets without compromising data privacy.

6.2.4 Scalability and Adaptability

Furthermore, our study highlights the scalability of federated learning models across diverse geographic settings. Despite regional disparities, federated learning models demonstrate adaptability and maintain robust anomaly detection performance across nodes. also proving the scalability

of this research, for more data from different locations to be added with the model expected to maintain competitive results.

6.3 Challenges in Model Generalization Across Diverse Regions

In our exploration of federated learning for anomaly detection across geographically diverse nodes, we encounter significant challenges in model generalization. Notably, integrating data from Oslo into our federated training setup with nodes in Tromsø and Gelio leads to marked performance discrepancies. This outcome shows the complexities involved in developing a single federated model that effectively handles data from distinct regional contexts.

Regional Data Variability

The integration of Oslo data introduces a skew in the performance of the global model. As a major urban center, Oslo likely exhibits different network traffic patterns and anomalies compared to the distributions found in northern locations like Tromsø and more remote regions like Gelio. This disparity results in a global model that, while performing adequately on data from Tromsø, struggled to adapt to the distinct characteristics of data from Oslo and Gelio.

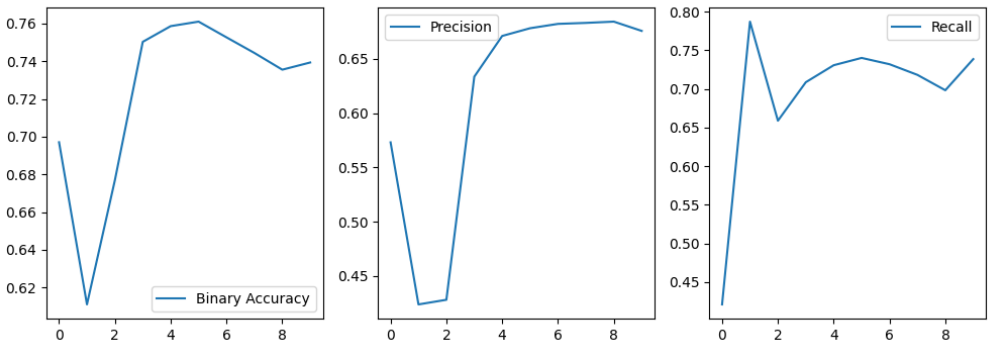


Figure 6.2: Best result achieved with Oslo, Gelio, Tromso combination with an Accuracy of 71% Precision of 73% and Recall of 68%

Generalization Difficulties

The node in Bud is used to replace the Oslo node. The model performs well with data from Tromsø, likely due to similar geographical and infrastructural characteristics shared with Bud. However, it struggled to adapt to data from Gelio, demonstrating a potential regional bias in learning.

The evaluation metrics reveal significant initial challenges, particularly in adapting to Gelio's data. This is represented by fluctuating loss that can also be seen in other nodes and low accuracy and

precision in early rounds. However, from Round 8 onwards, the model shows improvement, eventually reaching a peak accuracy of 75% and precision of 50% by Round 12 refer to (Figure 5.12). This improvement suggests that the federated learning model is able to adjust its parameters effectively over time, enhancing its ability to generalize across the different data distributions provided by each node.

6.3.1 Implications for Federated Learning

While federated learning excels at preserving data privacy and enabling localized processing, its success heavily depends on developing models that generalize effectively across diverse data sources, avoiding biases towards particular node characteristics.

The observed difficulty in balancing these aspects suggests that federated learning might benefit from node selection strategies. To enhance model generalization and performance:

Geographical Cohesion: Selecting nodes within similar geographic settings, such as all urban or all rural areas, could reduce variability in data characteristics that stem from regional differences. This approach might lead to models that are better tuned to specific environmental or infrastructural factors prevalent within these settings.

Regional Centralization: While federated learning is decentralized by design, introducing a level of centralization in terms of node selection could be beneficial. By grouping nodes from the same city or region, the model could leverage localized similarities in data patterns and anomalies, enhancing the overall model's effectiveness and efficiency.

6.4 Limitations and Further Work

During this study, several limitations were encountered.

One big limitation is the availability of computational resources. Given the high dimensionality of the data, only a subset is utilized for both centralized and federated learning approaches. As a result, the study may not fully capture the potential performance of these methods. With access to greater computational resources, it is possible that more comprehensive analyses can be conducted, potentially yielding improved results for both federated and centralized learning strategies.

Future research could also focus on implementing alternative weight aggregation methods, as discussed in the literature review chapter. Exploring these methods could enhance the performance and efficiency of federated learning models, particularly in scenarios with highly imbalanced and high-dimensional datasets.

Additionally, enhancing the privacy assurances within the federated learning architecture. Incorporating additional privacy-preserving techniques discussed also in the literature review of this study,

could improve data security and confidentiality.

Lastly, the integration of federated learning with split learning presents another avenue for future research. As highlighted in the literature review, split learning offers unique opportunities for collaborative model training while preserving data privacy. Investigating the combination of federated learning and split learning could lead to more novel approaches for secure and efficient model training in distributed environments.

Chapter 7

Conclusion

This thesis has demonstrated the effectiveness of federated learning for anomaly detection in mobile broadband networks, highlighting its potential to enhance security and efficiency without compromising data privacy. Through decentralized data processing and model training directly on devices, FL has shown superior performance in detecting network anomalies compared to traditional centralized methods. Despite these advancements, the research identified challenges such as data heterogeneity, scalability, and the need for robust privacy-preserving protocols, suggesting areas for further exploration and improvement.

Future research could focus on refining FL approaches by integrating advanced machine learning algorithms, enhancing data privacy measures, and improving system scalability. This thesis contributes to the ongoing development of machine learning in network management and opens new pathways for leveraging decentralized learning frameworks in complex, data-driven environments.

In light of these findings and their potential implications, it is my intention to submit this thesis as a research paper for publication. I believe that sharing these insights with the broader academic community in the form of a research paper will stimulate further research and discussion in this important field.

Bibliography

Abedi, A., & Khan, S. S. (2023). Fedsl: Federated split learning on distributed sequential data in recurrent neural networks. *Multimedia Tools and Applications*, 1–21.

Afroz, F., Subramanian, R., Heidary, R., Sandrasegaran, K., & Ahmed, S. (2015). Sinr, rsrp, rssi and rsrq measurements in long term evolution networks. *International Journal of Wireless & Mobile Networks*.

Aggarwal, C. (2023). The backpropagation algorithm. In *Neural networks and deep learning: A textbook* (pp. 29–71). Springer.

Ahasan, M. R., Haque, M. S., & Alam, M. G. R. (2022). Supervised learning based mobile network anomaly detection from key performance indicator (kpi) data. *2022 IEEE Region 10 Symposium (TENSYMP)*, 1–6.

Ahmed, M., & Pathan, A.-S. K. (2019). Investigating deep learning for collective anomaly detection-an experimental study. *Security in Computing and Communications: 6th International Symposium, SSCC 2018, Bangalore, India, September 19–22, 2018, Revised Selected Papers 6*, 211–219.

Alex, K. (2009). Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*.

Alghanmi, N., Alotaibi, R., & Buhari, S. M. (2022). Machine learning approaches for anomaly detection in iot: An overview and future research directions. *Wireless Personal Communications*, *122*(3), 2309–2324.

Almeida, A., Brás, S., Sargento, S., & Pinto, F. C. (2023). Time series big data: A survey on data stream frameworks, analysis and algorithms. *Journal of Big Data*, *10*(1), 83.

Attaran, M. (2023). The impact of 5g on the evolution of intelligent automation and industry digitization. *Journal of ambient intelligence and humanized computing*, *14*(5), 5977–5993.

Attaran, M., & Attaran, S. (2020). Digital transformation and economic contributions of 5g networks. *International Journal of Enterprise Information Systems (IJEIS)*, *16*(4), 58–79.

Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A. (2020). Usad: Unsupervised anomaly detection on multivariate time series. *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 3395–3404.

Benvenuto, D., Giovanetti, M., Vassallo, L., Angeletti, S., & Ciccozzi, M. (2020). Application of the arima model on the covid-2019 epidemic dataset. *Data in brief*, *29*, 105340.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., de Gusmão, P. P. B., et al. (2020). Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*.

Bhattacharya, A., Gawali, M., Seth, J., & Kulkarni, V. (2022). Application of federated learning in building a robust covid-19 chest x-ray classification model. *arXiv preprint arXiv:2204.10505*.

Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2021). A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, *54*(3), 1–33.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control*. John Wiley & Sons.

Bre, F., Gimenez, J., & Fachinotti, V. (2017). Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, *158*. https://doi.org/10.1016/j.enbuild.2017.11.045

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Choi, W., Kim, M.-J., Yum, M.-S., & Jeong, D.-H. (2022). Deep convolutional gated recurrent unit combined with attention mechanism to classify pre-ictal from interictal eeg with minimized number of channels. *Journal of Personalized Medicine*, *12*(5), 763.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Condoluci, M., & Mahmoodi, T. (2018). Softwarization and virtualization in 5g mobile networks: Benefits, trends and challenges. *Computer Networks*, *146*, 65–84.

Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. In *Machine learning techniques for multimedia: Case studies on organization and retrieval* (pp. 21–49). Springer.

Dai, Y., Chen, Z., Li, J., Heinecke, S., Sun, L., & Xu, R. (2023). Tackling data heterogeneity in federated learning with class prototypes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(6), 7314–7322.

Das, S., Tariq, A., Santos, T., Kantareddy, S. S., & Banerjee, I. (2023). Recurrent neural networks (rnns): Architectures, training tricks, and introduction to influential research. *Machine Learning for Brain Disorders*, 117–138.

De Turck, F., Kang, J.-M., Choo, H., Kim, M.-S., Choi, B.-Y., Badonnel, R., & Hong, J. W.-K. (2017). Softwarization of networks, clouds, and internet of things.

de Luca, A. B., Zhang, G., Chen, X., & Yu, Y. (2022). Mitigating data heterogeneity in federated learning with data augmentation. *arXiv preprint arXiv:2206.09979*.

ElMoaqet, H., Eid, M., Glos, M., Ryalat, M., & Penzel, T. (2020). Deep recurrent neural networks for automatic detection of sleep apnea from single channel respiration signals. *Sensors*, *20*(18), 5037.

El-Saleh, A. A., Alhammadi, A., Shayea, I., Alsharif, N., Alzahrani, N. M., Khalaf, O. I., & Aldhyani, T. H. (2022). Measuring and assessing performance of mobile broadband networks and future 5g trends. *Sustainability*, *14*(2), 829.

Feamster, N., & Rexford, J. (2017). Why (and how) networks should run themselves. *arXiv preprint arXiv:1710.11583*.

González, G. G., Tagliafico, S. M., Fernández, A., Gómez, G., Casas, P., et al. (2023). One model to find them all deep learning for multivariate time-series anomaly detection in mobile network data. *IEEE Transactions on Network and Service Management*.

Goudru, N., Puneeth, R., & Rao, K. P. N. (2021). Enhancement of performance of round-trip time using kalman filtering. *Advances in VLSI, Signal Processing, Power Electronics, IoT, Communication and Embedded Systems: Select Proceedings of VSPICE 2020*, 99–108.

Gupta, O., & Raskar, R. (2018). Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, *116*, 1–8.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107–116.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hua, X., Zhu, L., Zhang, S., Li, Z., Wang, S., Deng, C., Feng, J., Zhang, Z., & Wu, W. (2023). Genad: General unsupervised anomaly detection using multivariate time series for large-scale wireless base stations. *Electronics Letters*, *59*(1), e12683.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.

Isaksson, M., & Norrman, K. (2020). Secure federated learning in 5g mobile networks. *GLOBECOM 2020-2020 IEEE Global Communications Conference*, 1–6.

Kang, S., Ros, S., Song, I., Tam, P., Math, S., & Kim, S. (2023). Real-time prediction algorithm for intelligent edge networks with federated learning-based modeling. *framework*, *77*(2), 2023.

Kaur, J., Parmar, K. S., & Singh, S. (2023). Autoregressive models in environmental forecasting time series: A theoretical and application review. *Environmental Science and Pollution Research*, *30*(8), 19617–19641.

Ketkar, N., & Ketkar, N. (2017). Stochastic gradient descent. *Deep learning with Python: A hands-on introduction*, 113–132.

Kochetkova, I., Kushchazli, A., Burtseva, S., & Gorshenin, A. (2023). Short-term mobile network traffic forecasting using seasonal arima and holt-winters models. *Future Internet*, *15*(9), 290.

Konecnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, *8*.

Korsky, S. A., & Berwick, R. C. (2019). On the computational power of rnns. *arXiv preprint arXiv:1906.06349*.

Kremenova, I., & Gajdos, M. (2019). Decentralized networks: The future internet. *Mobile Networks and Applications*, *24*(6), 2016–2023.

Kvalbein, A., Baltrūnas, D., Evensen, K., Xiang, J., Elmokashfi, A., & Ferlin-Oliveira, S. (2014). The nornet edge platform for mobile broadband measurements. *Computer Networks*, *61*, 88–101.

Lederer, J. (2021). Activation functions in artificial neural networks: A systematic overview. *arXiv preprint arXiv:2101.09957*.

Li, Z., Li, Q., Zhou, Y., Zhong, W., Zhang, G., & Wu, C. (2023). Edge-cloud collaborative learning with federated and centralized features. *arXiv preprint arXiv:2304.05871*.

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A*, *379*(2194), 20200209.

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., & Miao, C. (2020). Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *22*(3), 2031–2063.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Lin, Z., Qu, G., Chen, X., & Huang, K. (2023). Split learning in 6g edge networks. *arXiv preprint arXiv:2306.12194*.

Liu, B., Lv, N., Guo, Y., & Li, Y. (2023). Recent advances on federated learning: A systematic survey. *arXiv preprint arXiv:2301.01299*.

Liu, L., & Si, Y.-W. (2022). 1d convolutional neural networks for chart pattern classification in financial time series. *The Journal of Supercomputing*, *78*(12), 14191–14214.

Liu, P., Xu, X., & Wang, W. (2022). Threats, attacks and defenses to federated learning: Issues, taxonomy and perspectives. *Cybersecurity*, *5*(1), 1–19.

Mandal, K., & Gong, G. (2019). Privfl: Practical privacy-preserving federated regressions on high-dimensional data over mobile networks. *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, 57–68.

Marfo, W., Tosh, D. K., & Moore, S. V. (2022). Network anomaly detection using federated learning. *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*, 484–489.

Martinez, G., Hernandez, J. A., Reviriego, P., & Reinheimer, P. (2023). Round trip time (rtt) delay in the internet: Analysis and trends. *IEEE Network*.

Matar, M., Xia, T., Huguenard, K., Huston, D., & Wshah, S. (2023). Multi-head attention based bi-lstm for anomaly detection in multivariate time-series of wsn. *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 1–5.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. *Artificial intelligence and statistics*, 1273–1282.

Meng, X., Wang, S., Liang, Z., Yao, D., Zhou, J., & Zhang, Y. (2021). Semi-supervised anomaly detection in dynamic communication networks. *Information Sciences*, *571*, 527–542.

Michelucci, U. (2022a). Feed-forward neural networks. In *Applied deep learning with tensorflow 2: Learn to implement advanced deep learning techniques with python* (pp. 61–109). Springer.

Michelucci, U. (2022b). An introduction to autoencoders. *arXiv preprint arXiv:2201.03898*.

Moustafa, N., & Slay, J. (2015). Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). *2015 military communications and information systems conference (MilCIS)*, 1–6.

Naidu, G., Zuva, T., & Sibanda, E. M. (2023). A review of evaluation metrics in machine learning algorithms. *Computer Science On-line Conference*, 15–25.

National Collegiate Cyber Defense Competition. (2024). *National Collegiate Cyber Defense Competition*. https://www.nationalccdc.org/

Nguyen, H. M., Chu, N. H., Nguyen, D. N., Hoang, D. T., Ha, M. H., & Dutkiewicz, E. (2022). Optimal privacy preserving in wireless federated learning system over mobile edge computing. *arXiv preprint arXiv:2211.07166*.

Nixon, C., Sedky, M., & Hassan, M. (2023). Salad: An exploration of split active learning based unsupervised network data stream anomaly detection using autoencoders. *Authorea Preprints*.

Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, *54*(2), 1–38.

Patel, M. B., Patel, J. N., & Bhilota, U. M. (2022). Comprehensive modelling of ann. In *Research anthology on artificial neural network applications* (pp. 31–40). IGI Global.

Pelati, A., Meo, M., & Dini, P. (2022). Traffic anomaly detection using deep semi-supervised learning at the mobile edge. *IEEE Transactions on Vehicular Technology*, *71*(8), 8919–8932.

Perifanis, V., Pavlidis, N., Koutsiamanis, R.-A., & Efraimidis, P. S. (2023). Federated learning for 5g base station traffic forecasting. *Computer Networks*, *235*, 109950.

Pfitzner, B., Steckhan, N., & Arnrich, B. (2021). Federated learning in a medical context: A systematic literature review. *ACM Transactions on Internet Technology (TOIT)*, *21*(2), 1–31.

Qian, Y., Hu, L., Chen, J., Guan, X., Hassan, M. M., & Alelaiwi, A. (2019). Privacy-aware service placement for mobile edge computing via federated learning. *Information Sciences*, *505*, 562–570.

Rajendra, S., Pradhan, C., & Kanniappan, J. (2023). An adaptive detection mechanism for iot devices anomalies using ai/ml based on user pattern. *Congress on Intelligent Systems*, 13–25.

Ren, J., Ni, W., Nie, G., & Tian, H. (2021). Research on resource allocation for efficient federated learning. *arXiv preprint arXiv:2104.09177*.

Samariya, D., & Thakkar, A. (2023). A comprehensive survey of anomaly detection algorithms. *Annals of Data Science*, *10*(3), 829–850.

Santana, J. A., Macías, E., Suárez, Á., Marrero, D., & Mena, V. (2017). Adaptive estimation of wifi rssi and its impact over advanced wireless services. *Mobile Networks and Applications*, *22*, 1100–1112.

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, *2*(3), 160.

Sarker, I. H., Hoque, M. M., Uddin, M. K., & Alsanoosy, T. (2021). Mobile data science and intelligent apps: Concepts, ai-based modeling and research directions. *Mobile Networks and Applications*, *26*(1), 285–303.

Sarsodia, T., Bhatt, U. R., & Upadhyay, R. (2021). Applications of rssi preprocessing in multi-domain wireless networks: A survey. *Advances in Computing and Network Communications: Proceedings of CoCoNet 2020, Volume 1*, 389–403.

Sater, R. A., & Hamza, A. B. (2021). A federated learning approach to anomaly detection in smart buildings. *ACM Transactions on Internet of Things*, *2*(4), 1–23.

Schmidt, R. M. (2019). Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*.

Sengupta, S., Kim, H., & Rexford, J. (2022). Continuous in-network round-trip time monitoring. *Proceedings of the ACM SIGCOMM 2022 Conference*, 473–485.

Shaheen, M., Farooq, M. S., Umer, T., & Kim, B.-S. (2022). Applications of federated learning; taxonomy, challenges, and research trends. *Electronics*, *11*(4), 670.

Shakir, Z., Mjhool, A. Y., Al-Thaedan, A., Al-Sabbagh, A., & Alsabah, R. (2023). Key performance indicators analysis for 4 g-lte cellular networks based on real measurements. *International Journal of Information Technology*, *15*(3), 1347–1355.

Sharafaldin, I., Lashkari, A. H., Ghorbani, A. A., et al. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, *1*, 108–116.

Sharafaldin, I., Lashkari, A. H., Hakak, S., & Ghorbani, A. A. (2019). Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. *2019 international carnahan conference on security technology (ICCST)*, 1–8.

Shawel, B. S., Debella, T. T., Tesfaye, G., Tefera, Y. Y., & Woldegebreal, D. H. (2020). Hybrid prediction model for mobile data traffic: A cluster-level approach. *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207655

Sheikhi, S., & Kostakos, P. (2023). Ddos attack detection using unsupervised federated learning for 5g networks and beyond. *2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 442–447.

Shen, L., Li, Z., & Kwok, J. (2020). Timeseries anomaly detection using temporal hierarchical one-class network. *Advances in Neural Information Processing Systems*, *33*, 13016–13026.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, *404*, 132306.

Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023). A comprehensive overview and comparative analysis on deep learning models: Cnn, rnn, lstm, gru. *arXiv preprint arXiv:2305.17473*.

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*.

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., & Pei, D. (2019). Robust anomaly detection for multivariate time series through stochastic recurrent neural network. *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2828–2837.

Sublime, J., & Kalinicheva, E. (2019). Automatic post-disaster damage mapping using deep-learning techniques for change detection: Case study of the tohoku tsunami. *Remote Sensing*, *11*(9), 1123.

Sutskever, I., Jozefowicz, R., Gregor, K., Rezende, D., Lillicrap, T., & Vinyals, O. (2015). Towards principled unsupervised learning. *arXiv preprint arXiv:1511.06440*.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, *105*(12), 2295–2329.

Taleb, T., Laghrissi, A., & Bensalem, D. E. (2020). Toward ml/ai-based prediction of mobile service usage in next-generation networks. *IEEE Network*, *34*(4), 106–111.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. *2009 IEEE symposium on computational intelligence for security and defense applications*, 1–6.

Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., & Chavez-Urbiola, E. A. (2023). Loss functions and metrics in deep learning. a review. *arXiv preprint arXiv:2307.02694*.

Thapa, C., Arachchige, P. C. M., Camtepe, S., & Sun, L. (2022). Splitfed: When federated learning meets split learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(8), 8485–8493.

Thudumu, S., Branch, P., Jin, J., & Singh, J. (2020). A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, *7*, 1–30.

Tian, Z., Zhuo, M., Liu, L., Chen, J., & Zhou, S. (2023). Anomaly detection using spatial and temporal information in multivariate time series. *Scientific Reports*, *13*(1), 4400.

Tuli, E. A., Lee, J. M., & Kim, D.-S. (2022). Federated learning-based computation offloading for low-bandwidth edge internet of things. *2022 27th Asia Pacific Conference on Communications (APCC)*, 377–379.

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine learning*, *109*(2), 373–440.

Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, *53*(8), 5929–5955.

Vepakomma, P., Gupta, O., Swedish, T., & Raskar, R. (2018). Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*.

Vucovich, M., Tarcar, A., Rebelo, P., Rahman, A., Nandakumar, D., Redino, C., Choi, K., Schiller, R., Bhattacharya, S., Veeramani, B., et al. (2023). Anomaly detection via federated learning. *2023 33rd International Telecommunication Networks and Applications Conference*, 259–266.

Wagner, B., & Cleland, K. (2023). Using autoregressive integrated moving average models for time series analysis of observational data. *bmj*, *383*.

Wang, S., Balarezo, J. F., Kandeepan, S., Al-Hourani, A., Chavez, K. G., & Rubinstein, B. (2021). Machine learning in network anomaly detection: A survey. *IEEE Access*, *9*, 152379–152396.

Wang, Z., Hu, Q., Xiong, Z., Liu, Y., & Niyato, D. (2023). Resource optimization for blockchain-based federated learning in mobile edge computing. *IEEE Internet of Things Journal*.

Wei, Y., Jang-Jaccard, J., Sabrina, F., Xu, W., Camtepe, S., & Dunmore, A. (2023). Reconstruction-based lstm-autoencoder for anomaly-based ddos attack detection over multivariate time-series data. *arXiv preprint arXiv:2305.09475*.

Xu, H., Ma, X., Wang, C., Wang, X., Xu, C., Gao, F., & Kong, L. (2022). A neural network approach for wireless spectrum anomaly detection in 5g-unlicensed network. *CCF Transactions on Pervasive Computing and Interaction*, *4*(4), 465–473.

Yang, M., & Zhang, J. (2023). Data anomaly detection in the internet of things: A review of current trends and research challenges. *International Journal of Advanced Computer Science and Applications*, *14*(9).

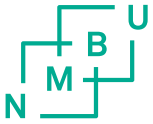
Zenke, F., & Neftci, E. O. (2020). Brain-inspired learning on neuromorphic substrates. *arXiv preprint arXiv:2010.11931*.

Zhang, H., Xiao, Z., Gu, J., & Liu, Y. (2023). A network anomaly detection algorithm based on semi-supervised learning and adaptive multiclass balancing. *The Journal of Supercomputing*, 1–36.

Zhao, Y., Chen, J., Guo, Q., Teng, J., & Wu, D. (2020). Network anomaly detection using federated learning and transfer learning. *International Conference on Security and Privacy in Digital Economy*, 219–231.

Zhou, Y., Shi, Y., Zhou, H., Wang, J., Fu, L., & Yang, Y. (2023). Toward scalable wireless federated learning: Challenges and solutions. *IEEE Internet of Things Magazine*, *6*(4), 10–16.

Zontou, E. (2023). Unveiling the evolution of mobile networks: From 1g to 7g. *arXiv preprint arXiv:2310.19195*.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway