Network Traffic Classification Under Label Scarcity and Privacy Constraints Using Federated Self-Supervised and Confident Learning

Ehsan Eslami

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

December 2025

© Ehsan Eslami, 2026

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Ehsan Eslami**

Entitled: **Network Traffic Classification Under Label Scarcity and Privacy Constraints Using Federated Self-Supervised and Confident Learning**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Name of the Chair

_____ External Examiner
Dr. Jamal Bentahar

_____ Examiner
Dr. Dongyu Qiu

_____ Supervisor
Dr. Walaa Hamouda

Approved by _____
Dr. Abdelwahab Hamou-Lhadj, Chair
Department of Electrical and Computer Engineering

_____ 2025

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Network Traffic Classification Under Label Scarcity and Privacy Constraints Using Federated Self-Supervised and Confident Learning

Ehsan Eslami

Network Traffic Classification (NTC) is essential for network management and security; however, it faces challenges such as label scarcity, pseudo-label noise, privacy concerns, non-independent and non-identically distributed (non-IID) data, and class imbalance in distributed environments. This thesis proposes two interconnected frameworks to address these issues. The centralized approach integrates traffic-adapted self-supervised learning (SSL) with confident learning (CL), employing a constraint-consistent autoencoder (AE) and enhanced tabular contrastive learning (TabCL) for robust pseudo-label generation from unlabeled flows, followed by CL-based denoising using per-class quantile thresholds, calibration-aware weighting, and balanced retention. This centralized SSL+CL pipeline is proven to be highly effective, establishing a strong performance benchmark.

The federated extension, FedSSL-NTC, distributes this process across clients, incorporating FedProx for non-IID stability, class-weighted losses for imbalance, sample-size-weighted (SSW) Federated Averaging (FedAvg) for fair aggregation, and a tailored secure aggregation (SecAgg) protocol for privacy preservation, ensuring no raw data sharing. Evaluations on a self-generated, unlabeled dataset combined with ISCX VPN-nonVPN and the UCDavis–QUIC benchmark demonstrate high effectiveness. Experimental results demonstrate that FedSSL-NTC achieves near-centralized accuracy. These contributions advance label-efficient, noise-robust, and privacy-aware NTC, bridging gaps in centralized and distributed paradigms for real-world applications in IoT, 5G/6G, and encrypted traffic scenarios.

Acknowledgments

First and foremost, I would like to express my gratitude to my supervisor, **Dr. Walaa Hamouda**, for his invaluable guidance, experience, and mentorship throughout this research. I am sincerely grateful for the confidence he placed in me and for the financial assistance provided. It is an honor to have completed my Master's under his supervision, and I am proud to be counted among his students. This thesis would not have been possible without his insightful advice and constant encouragement. I would also like to thank the members of my examination committee for agreeing to review this thesis and for their valuable time.

I would also like to extend my heartfelt thanks to my wife. I am deeply thankful for her patience and for always pushing me to stay motivated when I needed it most. I am also grateful to my family for their sacrifices and support of my decision to continue my academic path in Canada. Thank you all; I would not be here right now without you.

Contents

List of Figures

List of Tables

Nomenclature

AE	Autoencoder
BRC	Balanced Retention Constraint
CL	Confident Learning
CNNs	Convolutional Neural Networks
CSV	Comma-Separated Values
DL	Deep Learning
DPI	Deep Packet Inspection
FedAvg	Federated Averaging
FL	Federated Learning
GANs	Generative Adversarial Networks
GNNs	Graph Neural Networks
IoT	Internet of Thing
KNN	K Nearest Neighbors
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron

non-IID	non-Independent and non-Identically Distributed
NTC	Network Traffic Classification
QoS	Quality of Service
QUIC	Quick UDP Internet Connections
RNNs	Recurrent Neural Networks
SCE	Symmetric Cross-Entropy
SDN	Software-Defined Networking
SecAgg	Secure Aggregation
SSL	Self-supervised learning
SSW	Sample-size-weighted
SVM	Support Vector Machines
TabCL	Tabular Contrastive Learning

Chapter 1

Introduction

1.1 Network Traffic Classification

Network traffic classification (NTC) is a fundamental process in modern networking that refers to the task of identifying the application, service, or activity that generated network traffic by analyzing features obtainable from packet headers or aggregated flows [1]. The importance of NTC has surged in today's interconnected world [2], driven by the explosion of cloud computing, the rise of distributed services and Internet of Things (IoT) [3], the rollout of 5G networks, and the impending arrival of 6G technologies [4–6]. With global Internet users reaching over 4.1 billion in recent years and data creation projected to hit 181 zettabytes by 2025 [7], networks face unprecedented burdens. In modern networks, accurate NTC underpins traffic engineering, Quality of Service (QoS) [8], capacity planning, security monitoring, and anomaly detection. Consequently, high-accuracy NTC is indispensable in this landscape [9].

NTC encompasses various types, each tailored to specific objectives [1, 2, 9]. At a high level, a common goal is application identification (e.g., distinguishing streaming, messaging, gaming, or web applications) [10]. In security contexts, NTC enables intrusion/anomaly detection and malware/botnet traffic detection [11, 12]. Other fine-grained use cases include user-behavior inference, website fingerprinting, device identification, and quality-of-experience investigation [9].

Across these use cases, the classification target can be defined on packet-level inputs or on flow-level inputs. Packet-level classification examines individual packets, focusing on headers, payloads, or metadata like port numbers and packet sizes to infer immediate characteristics. Flow-level classification, on the other hand, aggregates sequences of packets sharing common attributes (such as source/destination IP addresses, ports, and protocols) into bidirectional flows, analyzing statistical/time-series features like counts, rates, flow duration, inter-packet intervals, and byte distributions [13, 14]. Packet-level classifications provide fine temporal detail but are costly to capture and store at scale and are increasingly content-agnostic due to encryption. Flow-level classifications are more scalable and privacy-conscious while still offering discriminative summary statistics and temporal descriptors. Flow-level approach is particularly useful for handling encrypted traffic, where payload inspection is infeasible [1, 9, 15].

Flow-level classification typically operates on a table of mixed feature types: continuous and categorical types. Continuous features, such as flow duration, total bytes and packets transferred, inter-arrival times, and packet rates, capture quantitative aspects of traffic behavior. The continuous features often exhibit algebraic interdependencies; for instance, average packet size equals total bytes divided by packet count, while throughput is derived as total bytes over flow duration. These relations should be preserved by the representation. Categorical features, including protocol type, IP version, and flow direction, are highly informative and equally vital for accurate classification, as they enable differentiation between traffic classes that continuous metrics alone might overlook; for example, UDP's stateless nature often signals real-time applications like streaming, distinct from TCP's reliable, or flow direction can helps to classify the download and upload traffics. Addressing these heterogeneous feature types and their inherent relationships demands traffic-adapted methods in NTC, which can enhance model robustness and scalability in diverse, encrypted environments [2].

1.2 Motivation

In NTC, Widespread encryption and protocol dynamism make legacy approaches (e.g., port-based identification or deep packet inspection (DPI)) unreliable or infeasible due to accuracy, privacy, and computational concerns [16, 17]. While AI-based methods can advance NTC by learning

complex patterns from data, they encounter significant challenges. Artificial intelligence methods, including machine learning (ML) and deep learning (DL), have become prominent in NTC due to their ability to extract patterns from statistical or sequential features without relying on traditional inspection techniques [18]. Supervised learning can achieve strong accuracy when abundant, diverse, and clean labels are available [19]. However, producing such labels is costly, time-consuming, and often impractical as protocols, applications, and traffic patterns evolve [20–23]. On the other hand, unsupervised learning discovers inherent structures in unlabeled data but often struggles to capture the complex patterns needed for precise traffic classification and typically underperforms supervised methods [8, 9]. Self-supervised learning (SSL) provides a middle path by exploiting large unlabeled corpora with pretext tasks and then using a small labeled set for fine-tuning; nevertheless, it introduces challenges in generating reliable pseudo-labels for downstream classification [24–26]. Pseudo-labels derived from SSL can be noisy due to imperfect pretext task performance, leading to degraded classifier accuracy if not explicitly controlled. This noise propagates errors, especially in imbalanced datasets where certain classes dominate. Furthermore, SSL representations may not fully capture traffic-specific constraints, such as heterogeneous feature types or class imbalances, exacerbating inaccuracies. In parallel, beyond labeling challenges, real deployments encounter additional constraints: network data are naturally distributed across sites/clients, making central aggregation of raw traffic data undesirable or impermissible due to privacy, regulatory requirements, and operational obstacles [27]; moreover, data across clients are non-independent and non-identically distributed (non-IID) and class-imbalanced across clients, which can bias training and degrade global accuracy if left unaddressed [28, 29]. Collectively, these issues necessitate a framework that simultaneously addresses all aforementioned challenges while maintaining high accuracy in NTC.

1.2.1 Label efficiency and robustness of pseudo-labels

In centralized settings, three practical issues motivate our first line of work. First, labeled data are scarce relative to the volume and diversity of unlabeled network traffic; second, SSL-derived pseudo-labels are imperfect and may contain systematic errors across classes; third, pseudo-label

noise interacts with calibration and imbalance, harming downstream training if not handled. Together, these factors motivate methods that (i) learn strong and traffic-adapted representations from unlabeled flows with tabular-appropriate pretext objectives and (ii) explicitly identify and attenuate noisy pseudo-labels before final training.

1.2.2 Privacy, non-IID data, and class imbalance in distributed settings

In realistic deployments, raw traffic often cannot be centralized. In distributed networks, privacy emerges as a critical concern, as centralizing raw traffic data exposes sensitive user information to breaches and regulatory violations. Federated learning (FL) enables collaborative training without sharing raw data [30], but it is not a complete privacy solution, and standard FL remains vulnerable to privacy risks. Even though raw data stays local, the model updates themselves can still leak sensitive information through gradient inversion or membership inference attacks [31]. This motivates the need for stronger privacy-enhancing defenses [31]. Protecting updates in FL typically relies on cryptographic or noise-based mechanisms, such as differential privacy [32], homomorphic encryption, or Secure Aggregation (SecAgg) [33]. In addition to these privacy vulnerabilities, FL brings its own distinct challenges: client data can be non-IID (feature and label skews), and class distributions across clients are frequently imbalanced [34]. These skews slow or destabilize training, bias models toward majority patterns, and reduce overall accuracy [35]. The motivation for our second line of work is thus to bring the robustness benefits of SSL and noise-aware pseudo-labeling into an FL setting while addressing the non-IID, imbalance, and privacy-leakage challenges intrinsic to federated NTC—without sacrificing the accuracy achieved by centralized training.

1.3 Approach Overview

This thesis presents two interconnected and complementary approaches to address the challenges in NTC, building on centralized and federated paradigms. The first framework operates in a centralized setting, assuming access to a large unlabeled dataset and a small labeled dataset of statistical and time-series features of traffic flows, to generate high-quality pseudo-labels for training a robust classifier. The second extends this to a federated environment, where data is distributed

across clients, emulating a real-world scenario that emphasizes privacy preservation while achieving comparable accuracy. Both approaches assume access only to header-derived packet or flow features (no payload inspection) and aim at multi-class application identification under realistic class imbalance. The methods are designed to be compatible with standard open datasets and with unlabeled traffic collected in practice.

1.3.1 Traffic-adapted Self-Supervised Learning and Confident Learning

In the centralized approach, the process begins with self-supervised pretraining on unlabeled traffic flows to learn meaningful representations. This involves two SSL branches tailored to tabular network-flow features: an Autoencoder (AE) tailored for constraint-consistent reconstruction, which handles mixed continuous and categorical flow features by preserving data integrity during encoding and decoding; and Tabular Contrastive Learning (TabCL) [36] enhanced with class-conditioned, constraint-preserving views and dual-head projections, allowing better differentiation of subtle traffic patterns through adaptive temperatures. These innovations make the methods traffic-adapted, improving representation quality for NTC by respecting flow-specific characteristics like heterogeneity and imbalance. Pseudo-labels are then generated by fine-tuning on a small labeled set and fusing predictions from both branches. Before training the final classifier as a downstream task, a traffic-adapted confident learning (CL) [37] module estimates label errors using out-of-sample probabilities, applying per-class quantile thresholds for calibration-aware reweighting and balanced retention to mitigate the impact of noisy pseudo labels. The result is a label-efficient training set that improves accuracy and robustness under scarce labels. A complete description of this centralized framework, together with experiments, is provided in Chapter 3.

1.3.2 Federated Self-Supervised Learning Under Privacy Constraints

Next, we extend the above idea to a federated setting to respect privacy constraints. The federated extension distributes the process across clients and a central server, assuming local unlabeled data per client and a small labeled seed on the server. Clients perform local self-supervised pretraining using the same traffic-adapted AE and TabCL. During pretraining, raw traffic remains on each client, and only learned model parameters are communicated to the server. After a brief

fine-tuning pass on a small labeled seed, clients generate pseudo-labels locally and apply the same confidence-aware noise handling. A federated training stage then learns the final classifier. Federated training incorporates proximal regularization [38] to stabilize updates amid non-IID data, class-weighted losses to counter imbalances, and sample-size-weighted (SSW) Federated Averaging (FedAvg) for aggregation to ensure fair contributions based on client data volumes. Furthermore, to address the privacy risks of update leakage, the framework integrates a tailored SecAgg [33] protocol, ensuring the server only learns the aggregated model updates without accessing individual client contributions. This enhances privacy in federated NTC while preserving convergence and accuracy. These adaptations make the framework robust for NTC in privacy-sensitive, heterogeneous environments, achieving near-centralized accuracy with reduced training time. We detail the architecture, training procedure, and evaluation of this approach in Chapter 4

1.4 Contributions

The key contributions of this thesis are as follows:

- We propose a novel centralized framework that integrates traffic-adapted SSL with CL for NTC, enabling high-accuracy classification by generating and refining pseudo-labels from large unlabeled datasets.
- We demonstrate the effectiveness of a two-branch SSL for pseudo-label generation: a constraint-consistent AE and enhanced TabCL with dual-head temperatures and constraint-preserving views, tailored to handle heterogeneous flow features and improve representation learning in traffic data. Their predictions are fused to create a high-coverage pseudo-labeled pool.
- We show that our traffic-adapted CL improves pseudo-label quality. This method estimates and attenuates pseudo-label noise using per-class quantile thresholds, logistic weighting, and balanced retention, effectively mitigating pseudo-label noise while retaining all data for robust training.
- We introduce a federated extension that enhances privacy by not sharing raw traffic, and designing a SecAgg protocol tailored to SSW FedAvg method to protect model updates.

- We demonstrate the robustness of the federated framework that addresses the non-IID data and class imbalance by using FedProx, class-weighted losses, and SSW FedAvg.

- We conduct a comprehensive evaluation across multiple datasets, including a self-generated unlabeled flow dataset, the ISCX VPN-nonVPN [14] benchmark, and the UCDavis–QUIC dataset [39], demonstrating superior accuracy and efficiency of both centralized and federated settings compared to state-of-the-art methods.

1.5 Thesis Organization

The remainder of this thesis is structured as follows. Chapter 2 reviews background and related work on NTC, including packet- and flow-level features, the limitations of legacy techniques, and learning-based approaches (supervised, unsupervised, and self-supervised, confident and contrastive learning), and introduces federated learning concepts relevant to NTC. Additionally, we highlight gaps addressed by our frameworks in this chapter. Chapter 3 details the centralized SSL + confidence-aware framework, covering methodology, model design, algorithms, theoretical foundations, data preparation, datasets, evaluation metrics, and experiments. Chapter 4 presents the federated framework, focusing on client–server workflow, its architecture, adaptations for privacy and heterogeneity, implementation details, experimental setup, and results. Finally, in the chapter 5 concludes with limitations, deployment considerations, and future research directions.

Chapter 2

Literature Review

This chapter surveys the foundations and prior research that motivate and contextualize the methods developed in this thesis. This chapter provides a comprehensive review of the existing literature relevant to NTC, with a focus on the evolution of methods from traditional techniques to advanced learning-based approaches. We begin by outlining the problem setting and traditional approaches and why they are increasingly ineffective in modern encrypted networks. Subsequent sections delve into ML and DL applications in NTC, including supervised and unsupervised paradigms, followed by an in-depth examination of SSL. Next, we introduce CL as a principled framework for coping with noisy labels, and finally we review FL for NTC, including works that combine SSL with FL. The chapter culminates in a synthesis and gap analysis, identifying how our proposed frameworks (traffic-adapted SSL with CL in both centralized and federated settings) address unresolved issues in the field.

2.1 Problem Setting and Traditional Methods for NTC

NTC aims to infer the application, service, or activity that produced the traffic based on observable features of packets or aggregated flows [9]. Packet-level classification analyzes individual packets, extracting features from headers, size, timing, or payloads to infer characteristics in real-time, while a flow-level view aggregates a bidirectional sequence of packets sharing the same 5-tuple (source/destination IPs, ports, protocol) and exposes statistical and time-series attributes (e.g.,

counts, duration, byte/packet rates, inter-arrival times) [15]. Flow-level features naturally comprise both continuous fields (e.g., total bytes, duration, average packet size, rates) and categorical fields (e.g., protocol, direction, cipher suite). In practice, packet payload inspection for NTC is increasingly infeasible due to the rise of encryption protocols like HTTPS and Quick UDP Internet Connections (QUIC), and the field has gravitated toward feature-based learning from observable metadata and statistics [18].

Traditional methods for NTC primarily include port-based identification and DPI [17]. Port-based approaches map traffic to applications using standardized port numbers assigned by the internet assigned numbers authority, but they falter with dynamic port usage and tunneling. DPI examines packet payloads for application-specific signatures, offering higher accuracy for unencrypted traffic [16]. However, DPI is computationally intensive, privacy-invasive, and ineffective against encrypted flows, which constitute over 90% of modern Internet traffic [8]. For example, [40] proposes BitProb, an application-identification method that builds probabilistic bit signatures from the first n payload bits of each bidirectional flow and represents them as a compact probabilistic counting deterministic automaton; a test flow is classified by comparing the probability of its n-bit string against learned signatures. Experiments on three datasets (one private and two public) report high accuracy. While less intrusive than full DPI, BitProb is still content-based and thus may face brittleness under encrypted traffic. Consequently, AI-based approaches, which can learn complex patterns from packet/flow features without relying on explicit rules or payload access, have become the dominant direction for NTC due to their adaptability and potential robustness under encryption in contemporary networks.

2.2 Machine Learning for NTC

In AI-driven approaches to NTC, ML plays a key role owing to its strong data-driven learning capabilities. ML involves algorithms that learn patterns from labeled data to make predictions or decisions without explicit programming. In NTC, ML models process flow or packet features to classify traffic without relying on payloads, using techniques like decision trees, random forests, support vector machines (SVM), and k-nearest neighbors (KNN) [41–45]. For instance, R. H. Serag

et al. [41] proposed an ML-based traffic classification method for Software-Defined Networking (SDN), demonstrating the effectiveness of classical ML algorithms like SVM and Logistic Regression in modern network architectures. [42] utilized supervised ML models such as Decision Tree and Random Forest in an SDN context for NTC, achieving high accuracy. Gomez et al. [45] proposed an ML-based system for classifying traffic flows in IP networks, focusing on early detection using the first flow and a dynamic threshold. They evaluated supervised models, including logistic regression, SVM, random forest, linear discriminant analysis, KNN, Gaussian Naive Bayes, and decision tree, on data captured via Wireshark, achieving a high classification rate. Despite their interpretability and efficiency, ML methods often remain constrained by the high labeling demand of supervised learning. ML approaches struggle with very complex patterns in high-dimensional traffic data, leading to suboptimal accuracy in encrypted or dynamic environments. This motivates the adoption of DL for more intricate feature representations.

2.3 Deep Learning for NTC

DL extends ML by using neural networks with multiple layers to automatically learn hierarchical features from data, enabling superior handling of complex, non-linear patterns in NTC. In the context of NTC, DL techniques are typically categorized into two groups: supervised and unsupervised methods.

2.3.1 Supervised Deep Learning Methods

At a high level, Supervised DL trains models on labeled data to minimize a loss function, such as cross-entropy, which measures the discrepancy between predicted and true labels. Weights in encoder layers learn features that are jointly optimized with a classifier head, and they are updated via backpropagation and optimization algorithms like stochastic gradient descent [46]. A foundational architecture is the multi-layer perceptron (MLP), consisting of input, hidden, and output layers where each neuron applies a non-linear activation (e.g., ReLU or Sigmoid) to weighted inputs, allowing the model to approximate arbitrary functions. For tabular flows, MLPs are common.

In addition to MLP, various supervised DL techniques can be applied to NTC, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Transformers, and Graph Neural Networks (GNNs), with some innovations and novelties, based on their suitability to packet/flow features [47–55].

In NTC, a CNN–LSTM autoencoder [47] that learns spatiotemporal flow features and classifies in latent space, with near-perfect accuracy on 1.42M flows (four classes) and robustness to imbalance. A CNN–RNN hybrid [48] achieved state-of-the-art results on CIC-IoT [56]. Lin et al. introduced IBNN, an information-bottleneck layer, in [49]. This method inserts an information-bottleneck layer into a CNN with a fully connected head, reaching 97.6% on the ISCX dataset; removing the IB layer drops performance by 8%, indicating better generalization. Lotfollahi et al. [50] proposed Deep Packet, combining stacked autoencoders and CNNs to classify encrypted traffic with 98% accuracy on ISCX VPN-nonVPN [14] while reducing manual feature engineering. Further supervised models include balanced supervised contrastive learning [51], which converts the first 25 packets to compact images and trains a ResNet-50 with supervised contrastive and imbalance-aware losses. TFE-GNN [52], a byte-level GNN-based temporal fusion encoder, reports state-of-the-art results, but assumes ample labeled data and does not address privacy-preserving training. Hu et al. [53] integrated LSTM, CNN, and squeeze-and-excitation modules, obtaining 98.8% average accuracy across multiple datasets and strong generalization. RBLJAN [54], which processes headers and payloads in parallel with byte–label joint attention and a GAN-based traffic generator, reporting 98% average F1 and 97% accuracy across datasets; and MTEFU [55], a multi-task setup (class/bandwidth/duration) sharing CNN, stacked autoencoder, and LSTM backbones with early-packet inputs, achieving 94% accuracy on UCDavis–QUIC [39] with only 150 labels. These supervised approaches often report more than 90% accuracy on benchmarks; however, these methods rely heavily on abundant, diverse, and clean labeled datasets, which are scarce and expensive to obtain in NTC due to traffic diversity and privacy concerns. They also suffer from overfitting to noisy labels and limited generalizability across encrypted protocols.

2.3.2 Unsupervised Deep Learning Methods

Unsupervised DL learns representations from unlabeled data by optimizing objectives like reconstruction or clustering, without explicit labels, to discover inherent data structures. Unsupervised learning methods have been investigated for NTC to avoid reliance on labeled datasets [57–59].

Gao et al. [58] present an unsupervised deep multi-source domain-adaptation model for NTC that integrates multi-scale feature extraction with adversarial, marginal, and conditional alignment mechanisms, together with a weighted, consistency-calibrated classifier; taken together, these components enable the model to achieve an accuracy of 89%. In a related line of work, [59] introduced a method that combines adversarial training with deep clustering for NTC. Using this approach, they reported a multi-class classification accuracy of 92.2%. Despite label independence, unsupervised methods generally have lower classification accuracy compared to supervised approaches on fine-grained application classes, especially under encryption and complex traffic dynamics [1, 9].

2.4 Self-Supervised Learning for NTC

This section explores SSL as a bridge between unsupervised and supervised paradigms, particularly suited for NTC, where labeled data is limited. We discuss the general SSL approach, followed by specific pretext tasks like autoencoders and contrastive learning, with a focus on tabular adaptations relevant to our frameworks.

2.4.1 Self-Supervised Learning Approach

SSL offers a middle ground: learn representations from large unlabeled data using pretext tasks (pretraining), then adapt the model with a small labeled seed (fine-tuning), and optionally generate pseudo-labels for additional supervision [60]. Pretraining in this manner gives the model a strong initialization that is already "familiar" with the data distribution. Instead of training from scratch on a small labeled set, fine-tuning only needs to adjust the pretrained features to the target label space. The key idea is that a well-chosen pretext task induces a supervisory signal from the data itself so that the encoder learns structure that is useful beyond the pretext objective [61]. Common pretext tasks include reconstruction (e.g., AEs), masking features, contrastive prediction (e.g.,

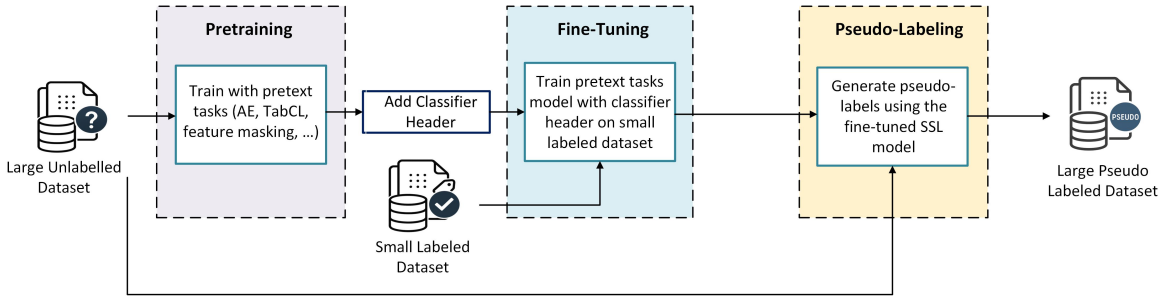


Figure 2.1: Overall Architecture of the Self-Supervised Learning approach.

pulling similar views close while pushing dissimilar ones apart), and generative modeling [62, 63]. With masking-based variants, selected features are hidden and the model must infer them from the remaining context, further pushing it to learn cross-feature dependencies [64]. Figure 2.1 shows the overall architecture of the SSL approach.

In summary, SSL decouples representation learning from label availability. Pretext tasks force the encoder to model meaningful structure in traffic flows (mixed continuous/categorical features and their relations). Fine-tuning then leverages a small labeled seed to specialize the representation for NTC. This approach is particularly appropriate in encrypted, fast-evolving network environments where labeled data are scarce but unlabeled traffic is abundant.

In NTC, SSL has been applied for traffic classification using different pretext methods and techniques [26, 65–73]. Towhid and Shahriar [26] applied SSL to encrypted NTC, attaining high accuracy with minimal labeled data across public datasets such as QUIC [39] and ISCX VPN-nonVPN [14]. ET-BERT [65] is a Transformer-based framework that learns contextualized datagram representations from large unlabeled traffic and then fine-tunes with few labels; it uses two traffic-tailored pretext tasks—Masked BURST Modeling and Same-origin BURST Prediction, supporting packet- and flow-level fine-tuning and achieving ¿96% macro-F1/accuracy. A notable example is YaTC [66], which couples a masked autoencoder with packet/flow-level Transformer attention over a multi-level flow representation byte layout; pretraining on unlabeled traffic and fine-tuning with few labels yields state-of-the-art results across multiple NTC datasets. Building on this direction, [67] proposes an SSL framework that uses the FlowPic method to convert flows into 2D histogram images, followed by contrastive representation learning with CNNs. In addition, [69]

presents a few-shot classifier leveraging autoencoders and deep graph convolutional networks. For SDN-based IoT NTC, [70] introduces a two-level fused network with self-adaptive manta ray foraging; incorporating SSL enhances feature learning and delivers robust performance. Bahlali et al. [71] develop a semi-supervised anomaly-detection framework for encrypted traffic, pairing an SSL masked-modeling pretext on unlabeled BiFlows with fine-tuning via a Custom-AE classifier using minimal labels. Yuan et al. [72] propose M3S-UPD, a multi-stage self-supervised unknown-aware packet detection framework that integrates semi-supervised learning with representation analysis via a four-phase iterative process, reaching 94.69% on ISCX. ET-SSL [73] by Sattar et al., which applies a contrastive SSL objective to flow statistics (e.g., packet lengths and inter-arrival times) to learn label-free representations for anomaly detection, achieving 96.8% accuracy. Collectively, these studies underscore SSL's ability to reduce reliance on labeled data, motivating our focus on SSL. Nonetheless, most SSL advances remain centralized, assuming pooled raw flows and overlooking privacy constraints and non-IID data. Moreover, Pseudo-labels derived from SSL can be noisy because of imperfect pretext, reducing classifier accuracy unless noise is explicitly controlled.

2.4.2 Autoencoders as a Pretext Task

Since AE serves as a pretext task in our NTC framework, we provide a focused explanation of its mechanics and operation. AEs are neural networks that learn an encoder–decoder pair that compresses and reconstructs inputs. An AE maps an input flow feature vector to a compact latent representation and then reconstructs the original input from that latent code. To succeed, the encoder must capture dependencies among features rather than memorize individual values [74]. In flow-level NTC, this means learning relationships among attributes (e.g., bytes, packets, duration, rates, protocol, direction) from unlabeled datasets for downstream classification. AEs serve as a pretext task by minimizing reconstruction loss, typically mean squared error for continuous features:

$$\mathcal{L}_{AE} = \frac{1}{N} \sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2, \tag{1}$$

where x_i is the input and $\hat{x}_i$ is the reconstruction. As a result, AEs provide general representations that can be fine-tuned and used to score unlabeled flows for pseudo-labeling. However,

original AEs may not preserve constraints (feature semantics and algebraic relations) and can produce miscalibrated pseudo-labels without additional noise-aware control. Accordingly, we propose a traffic-adapted AE that maintains semantic and algebraic consistency.

2.4.3 Tabular Contrastive Learning as a Pretext Task

Contrastive learning forms view pairs by augmenting an anchor sample (e.g., lightly perturbed but semantically consistent versions). It aims to learn representations by maximizing agreement between similar (positive) pairs and minimizing for dissimilar (negative) ones [63], often using the NT-Xent loss function:

$$\mathcal{L}_{NT-Xent} = -\log \frac{\exp(\text{sim}(v_i, v_j)/\tau)}{\sum_{[k \neq i]} \exp(\text{sim}(v_i, v_k)/\tau)}, \tag{2}$$

where $\tau > 0$ is temperature, v are projections, and sim is cosine similarity:

$$\text{sim}(v_i, v_j) = \frac{v_i.v_j}{|v_i||v_j|}. \tag{3}$$

Primarily developed for images, contrastive learning serves as a pretext task in SSL by creating augmented views (e.g., crops, rotation, flips). For traffic flows, replacements or perturbations must respect feature meaning. In NTC, contrastive SSL has been explored via FlowPic images [67].

On the other hand, TabCL adapts contrastive pretraining to tabular data. Standard table augmentations (random value swapping, feature drop) can break semantics. A recent approach [36] improves augmentation by class-conditioned corruption: when replacing a feature value for an anchor, sample the replacement from rows of the same class (estimated via pseudo-labels in a semi-supervised loop), thereby preserving class semantics in positive pairs. The method also studies correlation-based feature masking and uses a three-part network (encoder, pretrain head, classifier), with the pretrain head discarded after SSL. TabCL keeps the contrastive form above but changes the augmentation function to use class-conditioned replacements, improving positive-pair consistency. TabCL is suitable for NTC as flow features are tabular, and flow tables are mixed-type and class-imbalanced; class-conditioned replacements better preserve semantics than uniform column-wise

swaps, benefiting representation learning on traffic flows. However, TabCL relies on reliable pseudo-labels for conditioning; under severe class imbalance or calibration drift, view quality may degrade. In addition, the heterogeneous feature types in traffic data (categorical and continuous) complicate augmentation. Moreover, generic tabular augmentations may still ignore domain constraints (e.g., algebraic relations among continuous attributes), motivating traffic-adapted variants.

2.5 Confident Learning

A key gap in SSL is the noise in generated pseudo-labels, which can propagate errors during downstream training. Various methods mitigate this, such as label smoothing or noise estimation via filtering, reweighting, or robust losses. One effective approach is CL, which characterizes label errors under class-conditional noise assumptions. CL uses out-of-sample predicted probabilities and observed (noisy) labels; it estimates the joint distribution between noisy and latent clean labels under a class-conditional noise assumption. CL builds the confident joint by counting examples that exceed a per-class threshold and normalizes counts to estimate error rates; these statistics support pruning or weighting during training [37]. Formally, where $P(y = j; x)$ denotes the out-of-sample predicted probabilities for class j. For each class j, define a threshold t_j (e.g., mean of self-confidence for that class). The confident joint counts examples labeled as i but likely belonging to j when $P(j; x) \geq t_j$. Per-class thresholding improves robustness under class imbalance and heterogeneous calibration; subsequent pruning uses these estimates to reduce the impact of mislabeled samples during training. CL is model-agnostic and has been shown to improve learning with noisy labels across modalities. Original CL excels in identifying errors but assumes independent noise and may discard data aggressively, limiting its efficacy in imbalanced NTC, where class-conditional adaptations, calibration-aware weighting, and improving the per-class threshold are needed.

2.6 Federated Learning for NTC

FL allows multiple clients to collaboratively learn a global model without sharing raw traffic. Each round, clients train locally on their data and send model updates to a central server; the server

aggregates updates (e.g., via FedAvg: global weights as weighted average of client weights) and redistributes the model [30]. Benefits include privacy preservation, leveraging diverse data sources, and scalability [75]. Figure 2.2 summarizes the standard FL loop and the client–server workflow.

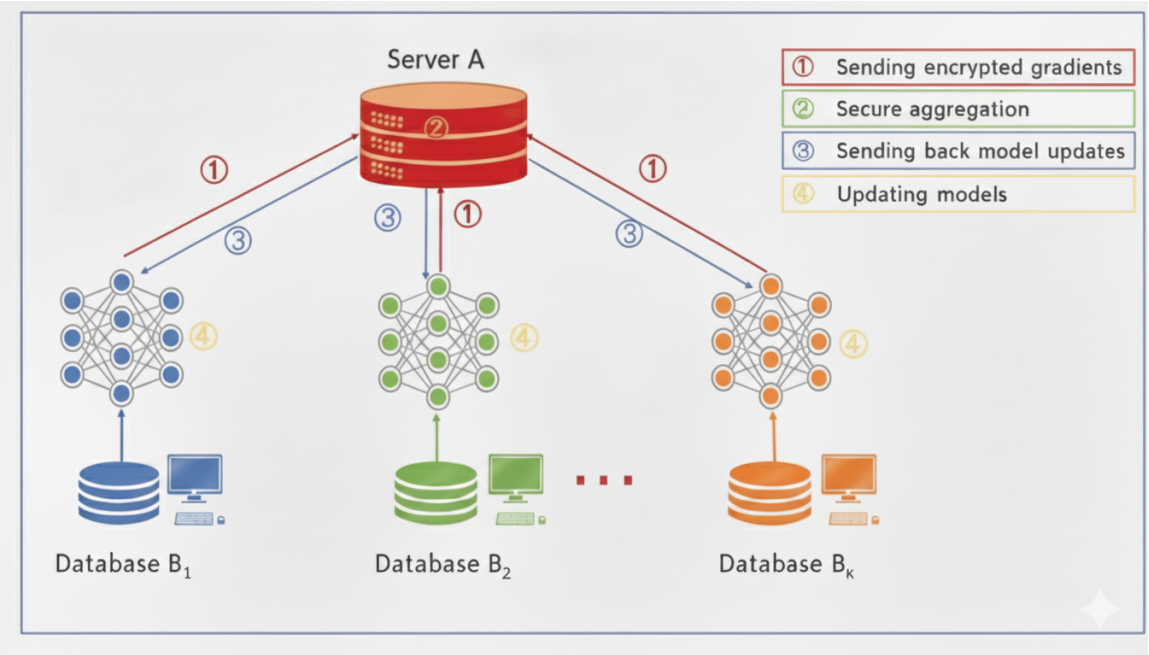


Figure 2.2: Standard Federated Learning Overview.

Given the distributed nature of network infrastructure and network traffic that contains sensitive user information, centralizing raw traces for model training is not always feasible. FL offers an appropriate solution for NTC [28, 76–78]. For example, Tran et al. [28] presented an incremental knowledge consolidation framework, integrating BERT-based embeddings and consolidated attention maps to support adaptive class-incremental learning under non-IID traffic. [76] developed a Byzantine-resilient FL algorithm (B-RLSFedAvg) for traffic classification that selects gradients least likely to be adversarial via pairwise similarity, mitigating arbitrary updates without prior Byzantine knowledge. C. Qiu introduced NTFLELM [77], which combines FL with extreme learning machines and achieves strong NTC performance. Additionally, Z. Jin et al. [78] proposed FedETC, a federated framework for encrypted traffic classification built on 1D CNNs that removes manual feature engineering and attains accuracy comparable to centralized training on real-world data. While these FL approaches offer privacy protection, they often rely on sufficient per-client

labeled data, which restricts applicability under label scarcity.

Class imbalance (biasing toward majority classes) and non-IID data distributions (feature/label skews causing divergence) are significant challenges in FL [79, 80]. This is especially true for NTC, where some application classes (e.g., video streaming) tend to dominate, and client datasets differ due to network heterogeneity [81]. Wang et al. [82] mitigated class imbalance in mobile traffic classification by employing weighted loss functions, improving fairness for underrepresented classes. Jimenez-Gutierrez et al. [35] offered a comprehensive empirical study of non-IID effects in federated learning, quantifying label, feature, quantity, and spatiotemporal skews via Hellingeri distance and benchmarking strategies such as FedProx [38] across multiple datasets. While these techniques enhance FL robustness, they have been tailored only rarely specifically to NTC. [83] proposed a Federated Class Incremental Learning (FCIL) framework to address the challenge of ”catastrophic forgetting,” where NTC models become outdated as new applications appear.

As mentioned, the standard FL loop is vulnerable to privacy risks, as model updates can be exploited by an honest-but-curious server to infer information about a client's local data. Regarding privacy-enhancing in FL for NTC, Yin et al. [84] propose FLSN-MVO, a privacy-aware edge FL framework that pairs a Siamese network with Multi-Verse Optimization and adaptive weighted aggregation to mitigate data-quality heterogeneity and accelerate convergence. They also add a key-distribution–based encryption mechanism for uploading model updates. The model was validated on intrusion detection system datasets. To harden privacy in encrypted-traffic analytics, in [85], a secure federated distillation design uses two non-colluding servers and secret-shared logits, reducing leakage while retaining.

Combining SSL and FL provides a promising solution to the dual challenges of limited labels and privacy in NTC. SSL can localize representation learning (clients pretrain on unlabeled flows) and use a small labeled seed on the server or clients for fine-tuning; pseudo-labels are then generated locally and used to train a federated classifier. This design respects privacy (only parameters move) while improving label efficiency. [86] proposed a federated semi-supervised NTC framework in which clients perform unsupervised pretraining and the server fine-tunes on a small labeled set via FedAvg. For smart-home traffic, Z. Wang et al. [87] use autoencoders to create pseudo-labels and federated training across edge gateways, lowering label needs but leaving pseudo-label noise

unaddressed. [88] presented FC-RTC, a robust federated contrastive approach for low-quality data (packet loss, label noise, skew) using supervised contrastive loss for representation learning and FedAvg for aggregation. Xiao et al. [89] developed FS-GAN, a federated SSL framework for traffic synthesis and classification at the edge using Generative Adversarial Networks (GANs), reporting 89% accuracy. FCAL [90] integrates adversarial AE representation learning with contrastive objectives under FL, plus asynchronous aggregation to reduce waiting. While the framework effectively combines SSL and FL to handle label scarcity, it does not explicitly address FL-specific challenges like non-IID data distributions or class imbalances. C. Lin et al. [91] proposed a federated semi-supervised framework that trains unsupervised encoders across clients and a server-side classifier; across single- and multi-packet features. This framework does not propose specific mechanisms to address non-IID distributions or class imbalance. FedEdge AI-TC [92] combines FL with semi-supervised variational AE-CNN and model pruning/interpretability for edge gateways; While this approach tackles both label scarcity and privacy, it uses a pre-train/fine-tune model rather than pseudo-labeling, and thus does not include a mechanism for pseudo-label denoising. Together, these studies underscore the promise of combining SSL and FL. While promising, many do not explicitly address pseudo-label noise, robust privacy-enhancing mechanisms beyond basic data locality, or simultaneously tackle non-IID and class imbalance in a unified way.

2.7 Synthesis and Gap Analysis

The literature reveals four persistent gaps: (i) label scarcity in supervised DL; (ii) lower precision of unsupervised methods for fine-grained, encrypted traffic; (iii) pseudo-label noise in SSL pipelines; and (iv) privacy-leakage from model updates plus distributional challenges (non-IID/class imbalance) in realistic deployments. Methods like TabCL improve tabular pretraining, and CL provides a principled denoising mechanism, but prior work rarely integrates the two with explicit domain adaptations for traffic flows, nor combines them with an FL recipe that addresses privacy and non-IID/imbalance jointly. A detailed comparison of our proposed framework against the representative related works, is provided in Table 2.1 that visually summarizes the gaps addressed.

Table 2.1: Comparison of proposed framework with Representative Related Works

Category	Work	Architecture	Label Scarcity	Pseudo-Label Denoise	Privacy	FL Robustness (Non-IID/Imbalance)
Supervised Learning	[41–45], [47–54]	Centralized	✗	N/A	✗	N/A
	[55]	Centralized	◐	N/A	✗	N/A
Self-Supervised Learning	[26, 65–73].	Centralized	✓	✗	✗	N/A
	[93] (chapter 3)	Centralized	✓	✓	✗	N/A
Federated Learning	[76–78]	Federated	✗	N/A	◐	✗
	[28, 35, 82, 83]	Federated	✗	N/A	◐	✓
	[34, 84]	Federated	✗	N/A	✓	✓
	[85]	Federated	✗	N/A	✓	✗
SSL + FL	[86, 87, 90–92]	Federated	✓	✗	◐	✗
	[88]	Federated	✗	✗	◐	✓
	[89]	Federated	✓	✗	◐	✓
This Work	**FedSSL-NTC (Ours)**	**Federated**	✓	✓	✓	✓

✓ = Addressed ✗ = Not Addressed ◐= Partially Addressed N/A = Not Applicable

This thesis addresses these gaps by (a) coupling traffic-adapted SSL via AE reconstruction tailored to mixed continuous/categorical flow features and TabCL with class-conditioned, constraint-preserving views, with (b) a traffic-adapted CL stage to attenuate pseudo-label noise using per-class quantile thresholds, logistic weights for smooth attenuation, and balanced retention guided by the confident joint, and (c) a federated extension that enhances privacy at two levels: by keeping data local and by using tailored SecAgg, while stabilizing learning under non-IID and class imbalance. Detailed methodologies and evaluations appear in Chapters 3 and 4.

Chapter 3

Centralized Network Traffic Classification Using Self-Supervised Learning and Confident Learning

3.1 Introduction

Building on the foundational concepts and motivations outlined in Chapter 1, where we introduced the challenges of NTC in modern networks, we highlighted the need for label-efficient, robust methods, and based on the comprehensive literature review in Chapter 2, which surveyed traditional, ML, DL, SSL, and CL approaches while identifying persistent gaps in label scarcity, limited precision of unsupervised methods, and pseudo-label noise, this chapter focuses on the centralized framework for NTC. Here, we present a detailed methodology that integrates traffic-adapted SSL for pseudo-label generation with an enhanced CL module to mitigate noise, enabling high-accuracy classification under limited labeled data. This chapter is based on material of our paper that was previously published in the IEEE Open Journal of the Communications Society (OJCOMS) [93].

This chapter develops our centralized approach for NTC. We first formalize the problem, assumptions, and notation. We then detail a two-branch SSL pipeline for pseudo-labeling from a large unlabeled pool, including the AE and TabCL branches, along with their fusion. Next,

we introduce a traffic-adapted CL stage that denoises pseudo-labels via per-class quantile thresholds, calibration-aware logistic weighting, and balanced retention, followed by the final classifier trained with a weighted Symmetric Cross-Entropy (SCE) loss. Finally, we conclude with a comprehensive experimental evaluation across three datasets (self-generated, ISCX VPN–nonVPN, UC-Davis–QUIC), ablations, compute profile, and comparisons to recent state-of-the-art methods.

3.2 Problem Statement, Assumptions, and Notation

In our proposed framework, we address multi-class NTC over K application classes using flow-level, header-derived heterogeneous features (no payload inspection). Let the small labeled set be defined:

$$D_s = \{(\mathbf{x}_i, y_i)\}_{i=1}^{|D_s|}, \quad y_i \in \{1, \ldots, K\}, \tag{4}$$

and the large unlabeled set drawn from the same deployment domain:

$$D_l = \{\mathbf{x}_j\}_{j=1}^{|D_l|}. \tag{5}$$

Each flow $\mathbf{x} \in R^d$ includes heterogeneous attributes: continuous features $\mathbf{x}_{\text{cont}}$ (e.g., counts, bytes, duration, rates) and categorical features $\mathbf{x}_{\text{cat}}$ (e.g., protocol, direction), numerically encoded. Continuous attributes encompass statistical features (e.g., total packets, total bytes, flow duration, average packet size) and time-series features (e.g., inter-arrival times, burst patterns), which are crucial for classification as they reflect aggregate behavior and temporal dynamics of traffic.

The goal is to train a multiclass classifier $\mathcal{C}_{\text{final}} : R^d \to \{1, \ldots, K\}$ that achieves high accuracy and macro-F1 with scarce labels by utilizing D_l through self-supervision and noise-aware pseudo-labeling. Specifically, pseudo-labels for D_l are generated via two complementary SSL branches adapted for NTC: an AE with constraint-consistent reconstruction for heterogeneous features, and TabCL with class-conditioned, constraint-preserving augmentations and dual-head projections. Predictions from both branches are fused using a confidence/margin voting rule when available. To handle noise, a traffic-adapted confident learning module is applied, followed by training $\mathcal{C}_{\text{final}}$ on the weighted pseudo-labeled set.

We assume a closed-world setting with K known classes, class imbalance, and calibration drift typical in real NTC, with pseudo-labels from SSL containing noise. Features are derived from flows/headers only (no payload inspection). The focus is on robustness to label scarcity and pseudo-label noise. Table 3.1 summarizes key notations used in this chapter.

Table 3.1: List of Notations.

Symbol	Description
$\mathbf{x}$	Flow feature vector in $\mathbf{R}^d$
$\mathbf{x}_{\text{cont}}$, $\mathbf{x}_{\text{cat}}$	Continuous / categorical feature slices
y, $\tilde{y}$	True label, pseudo-label
D_s, D_l	Small labeled dataset, Large unlabeled dataset
K	Number of application classes
d	Feature dimension
$\mathcal{E}$, $\mathcal{D}$, $\mathcal{C}$, $\mathcal{P}$	Encoder, decoder, classifier, projection head (TabCL)
$\widehat{\mathbf{x}}$	Autoencoder reconstruction of $\mathbf{x}$
$\mathcal{L}$, θ, η	Loss function, model parameters, learning rate
$\mathcal{G}$, g	Set of algebraic constraints, residual function
ϕ	Weights for constraint residuals in AE
$\widetilde{\mathbf{x}}_i^{(t)}$	replaced view in TabCL
$\breve{\mathbf{x}}_i^{(t)}$	Adjusted view by constraint-projection
r, τ	TabCL replacement rate, Temperature (TabCL)
$\mathbf{v}_i^{(t)}$	Contrastive projection vector
N, F	Mini-batch size, number of folds for CV
λ	Mixing coefficient in final TabCL loss
$\mathbf{p}$	Predicted probability vector
$\widehat{c}(\mathbf{x})$	fused pseudo-label (final vote)
$m(\mathbf{x})$	Confidence-margins (top-1 minus runner-up)
s_i	Self-confidence for sample i
$\mathcal{S}_j$	Self-confidence set for class j
$t_j^{(q)}$, q	q-quantile threshold for class j; quantile level
σ_j, γ	robust scale for class j, slope
w_i, $w_{\min}$	Calibration-aware per-sample weight, minimum weight for CL
$\widehat{Q}$	Confident joint matrix
ρ_j, n_j	estimated clean fraction for class j, Count in class j
T_j, M_j, a_j	Target mass; current mass; class scaling factor
w_i'	Final per-sample weight
$\text{clip}(x, \ell, u)$	Clipping operator $\min(\max(x, \ell),\, u)$
α, β	SCE loss coefficients (forward/reverse CE)

3.3 Self-Supervised Learning for Pseudo-Labeling

This section details the SSL stage that converts D_l into a pseudo-labeled pool. We first describe the system model and data flow, then present two complementary SSL branches (AE and TabCL), followed by a lightweight confidence–margin fusion.

3.3.1 System Model and End-to-End Data Flow

Raw packet traces are captured passively at monitored links and aggregated into bidirectional flows using the 5-tuple (source IP, destination IP, source port, destination port, protocol), with idle/active timeouts applied. Long connections may be segmented via time windows for memory efficiency. Flows are preprocessed to extract features, categorized as statistical (e.g., total packets, total bytes, flow duration, average packet size) and time-series (e.g., inter-arrival times, burst patterns). These features summarize flow characteristics and temporal patterns, aiding in distinguishing application types—such as video streaming's larger byte volumes versus chat applications' smaller ones, or real-time traffic's specific timing needs.

Beyond the primary features, secondary features are derived through aggregations and ratios (e.g., mean/variance of inter-arrival times, byte/packet rates, up/down throughput, burst-length ratios), enriching the tabular representation without payload access. In addition to these numeric attributes, categorical features (e.g., protocol, direction) are encoded as one-hot or embeddings, while numeric fields are standardized/normalized. This yields a large unlabeled traffic-flow dataset D_l; a small labeled subset D_s is obtained via DPI, heuristics, or manual verification, used solely for fine-tuning and selection. The unlabeled dataset generation pipeline from raw packets is illustrated in Figure 3.1.

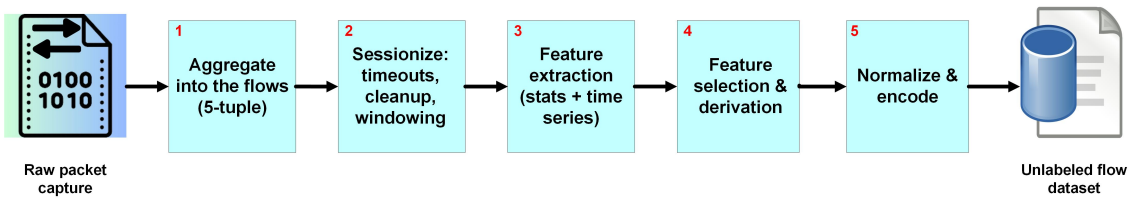


Figure 3.1: Pipeline from raw packet traces to an unlabeled traffic-flow feature dataset.

As depicted in Figure 3.2, the framework uses two independent SSL paths to pseudo-label D_l: an AE reconstructing heterogeneous features and a TabCL with constraint-preserving views and dual-head projections specialized for continuous vs. categorical slices. After pretraining, each branch replaces its decoder/projection with a classifier and fine-tunes on D_s (freeze-unfreeze). Each method can produce pseudo-labels alone. Using two SSL methods is advantageous because the AE emphasizes reconstruction fidelity, while TabCL focuses on discriminative invariances and considers heterogeneity through its dual heads, capturing complementary patterns. When both are available, predictions fuse via confidence/margin voting to enhance pseudo-ables quality. alternatively, the single-branch output is used. The pseudo-labeled set then proceeds to traffic-adapted CL for noise mitigation before final classification. This overview sets the stage; detailed descriptions, algorithms, and mathematical formulas for NTC-aware AE, TabCL, and fusion follow in the next separate subsections.

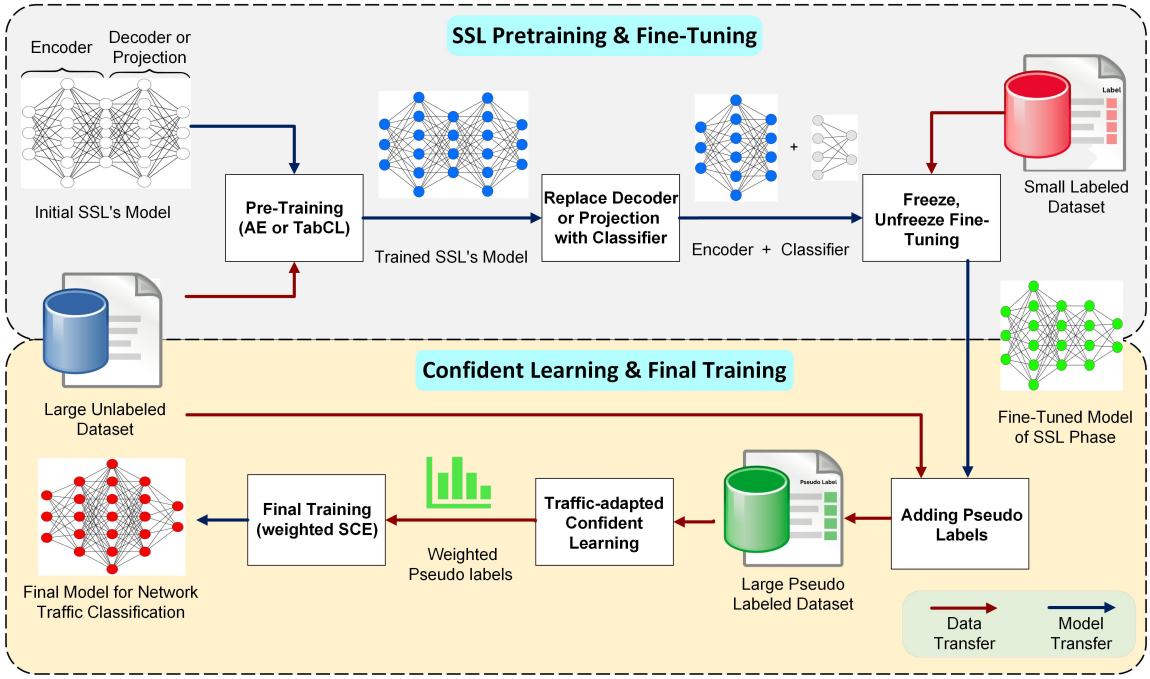


Figure 3.2: Architectural components of traffic classification with SSL and CL.

3.3.2 Autoencoder-based Self-supervised Learning

Autoencoders learn compressed representations by encoding inputs to a lower-dimensional latent space and reconstructing them, capturing patterns unsupervised. We represent each flow as a feature vector $\mathbf{x} \in R^d$ with continuous ($\mathbf{x}_{\text{cont}}$) and categorical ($\mathbf{x}_{\text{cat}}$) attributes. Categorical attributes are encoded in standard numeric form (e.g., one-hot or embeddings) and are reconstructed with a cross-entropy objective. This heterogeneous structure is characteristic of tabular network flow representations and motivates losses respecting numeric fidelity and categorical consistency.

Pretraining with Constraint-Consistent Reconstruction

Let $\mathcal{E}(\mathbf{x};\theta_{\mathcal{E}})$ and $\mathcal{D}(\mathbf{h};\theta_{\mathcal{D}})$ denote the encoder and decoder. For input $\mathbf{x}$, latent $\mathbf{h} = \mathcal{E}(\mathbf{x})$ and reconstruction $\widehat{\mathbf{x}} = \mathcal{D}(\mathbf{h})$ with splits $\widehat{\mathbf{x}}_{\text{cont}}$ and $\widehat{\mathbf{x}}_{\text{cat}}$. The pretraining objective combines mixed reconstruction with a soft penalty that nudges simple algebraic identities among continuous features:

$$\mathcal{L}_{\text{AE}} \;=\; \underbrace{\text{MSE}\big(\mathbf{x}_{\text{cont}}, \widehat{\mathbf{x}}_{\text{cont}}\big) + \text{CE}_{\text{cat}}\big(\mathbf{x}_{\text{cat}}, \widehat{\mathbf{x}}_{\text{cat}}\big)}_{\text{mixed reconstruction}} \;+\; \underbrace{\mathcal{L}_{\text{cons}}}_{\substack{\text{constraint}\\ \text{consistency}}}, \tag{6}$$

where MSE is mean squared error on continuous features, and CE sums cross-entropy over categorical fields (softmax-modeled). Traffic data features algebraic relations (e.g., rate = bytes / duration; packet length = payload + header). When the AE reconstructs such tuples, encouraging these relations to hold improves plausibility and stabilizes the latent representation. We formalize this with a set of algebraic residuals applied to the reconstructed continuous vector $\widehat{\mathbf{x}}_{\text{cont}}$:

$$\mathcal{G} \;=\; \big\{\, g_m : R^{d_{\text{cont}}} \to R \;\mid\; m = 1, \ldots, |\mathcal{G}| \,\big\}, \tag{7}$$

where each $g_m(\widehat{\mathbf{x}}_{\text{cont}})$ is the residual of constraint m when evaluated on the reconstructed continuous vector. We then define

$$\mathcal{L}_{\text{cons}} \;=\; \sum_{m=1}^{|\mathcal{G}|} \phi_m \,\big\|\, g_m\big(\widehat{\mathbf{x}}_{\text{cont}}\big) \,\big\|_1, \tag{8}$$

where $\phi_m > 0$ are modest weights. This penalty nudges the encoder and decoder to reconstruct plausible tuples without prescribing any particular feature identity, and improves the semantic quality of the representation reused at fine-tuning time.

Fine-Tuning and Pseudo-Labeling

Post-pretraining on D_l with (6), we replace $\mathcal{D}$ by a classifier $\mathcal{C}(\cdot;\theta_{\mathcal{C}})$, apply a short freeze–unfreeze schedule. This schedule is a standard practice that reduces optimization drift when transplanting a new head on a pretrained encoder. First, we freeze $\theta_{\mathcal{E}}$ and optimize $\theta_{\mathcal{C}}$ on D_s to stabilize the new head, then jointly unfreeze $(\theta_{\mathcal{E}}, \theta_{\mathcal{C}})$ and fine-tune on D_s with the standard cross-entropy:

$$\mathcal{L}_{\text{CE}} \;=\; -\frac{1}{|D_s|} \sum_{(\mathbf{x},y)\in D_s} \log\Big(\mathcal{C}\big(\mathcal{E}(\mathbf{x})\big)[y]\Big). \tag{9}$$

After fine-tuning, we apply the trained head to all $\mathbf{x} \in D_l$ and assign pseudo-labels by top-1 prediction.

Hyperparameter Selection for AE

Latent width, learning rates, freeze duration, and ϕ_m are selected by stratified cross-validation on D_s, targeting macro-F1 post-fine-tuning, keeping constraints modest and adaptive. Algorithm 1 summarizes the procedure of AE in our framework.

Algorithm 1 Autoencoder-SSL with Constraint-Consistent Reconstruction

Require: Unlabeled D_l, labeled D_s, batch size m, learning rate η
Initialize encoder $\mathcal{E}(\cdot;\theta_{\mathcal{E}})$, decoder $\mathcal{D}(\cdot;\theta_{\mathcal{D}})$
Pretrain on D_l:
for epoch $= 1, 2, \ldots$ **do**
  for mini-batch $\{\mathbf{x}_i\}_{i=1}^{m} \subset D_l$ **do**
    $\mathbf{z}_i \leftarrow \mathcal{E}(\mathbf{x}_i)$; $\widehat{\mathbf{x}}_i \leftarrow \mathcal{D}(\mathbf{z}_i)$
    Compute $\mathcal{L}_{\text{AE}}$ from (6) with $\mathcal{L}_{\text{cons}}$ in (8); update $(\theta_{\mathcal{E}}, \theta_{\mathcal{D}}) \leftarrow (\theta_{\mathcal{E}}, \theta_{\mathcal{D}}) - \eta\nabla\mathcal{L}_{\text{AE}}$.
  end for
end for
Replace $\mathcal{D}$ by classifier $\mathcal{C}(\cdot;\theta_{\mathcal{C}})$
Freeze $\theta_{\mathcal{E}}$; train $\theta_{\mathcal{C}}$ on D_s using CE (9)
Unfreeze $(\theta_{\mathcal{E}}, \theta_{\mathcal{C}})$; joint fine-tuning on D_s with (9)
Pseudo-labeling: for all $\mathbf{x} \in D_l$, set $\widehat{c} \leftarrow \arg\max_k \mathcal{C}(\mathcal{E}(\mathbf{x}))[k]$
Ensure: Pseudo-labeled $D_l = \{(\mathbf{x}, \widehat{c})\}$

3.3.3 Tabular Contrastive Learning (TabCL)

TabCL offers an alternative SSL approach. TabCL learns invariant representations via contrastive loss, effective for tabular data by maximizing positive-pair similarity and distinguishing negatives. The workflow is presented in Figure 3.3. We adopt the standard two-view NT-Xent setup and tailor it to network-traffic data with two components: a class-conditioned replacement scheme to create views, and a constraint-preserving projection that enforces simple identities among reconstructed continuous attributes in network traffic datasets. In addition, we use a dual-head contrastive objective to accommodate heterogeneous feature types with distinct temperature scales. This TabCL branch is independent of AE and can be run alone to generate the pseudo-labeled pool.

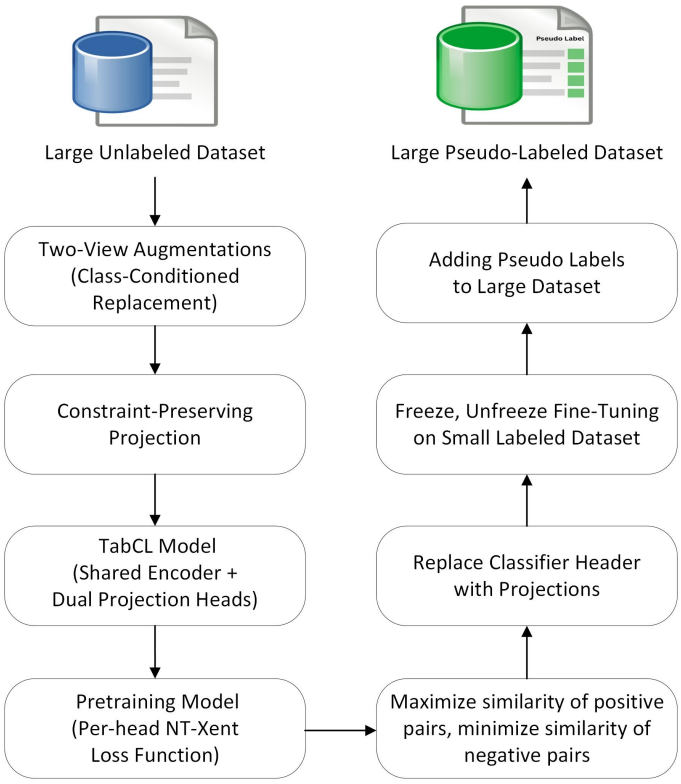


Figure 3.3: Workflow of TabCL for pseudo-labeling.

Class-Conditioned Two-View Augmentations.

Before pretraining in TabCL, we first obtain primary pseudo-labels $\{\tilde{y}_i^{(0)}\}$ for D_l by training a light classifier on D_s and applying it to D_l. During contrastive pretraining, we periodically refresh

these pseudo-labels: at predefined intervals, we freeze the current encoder $\mathcal{E}$, train a fresh linear probe on $\{\mathcal{E}(\mathbf{x}), y\}_{(\mathbf{x},y)\in D_s}$, and reclassify $\{\mathcal{E}(\mathbf{x})\}_{\mathbf{x}\in D_l}$ to obtain $\{\tilde{y}_i^{(t)}\}$. Refreshing stops after a small number of rounds or once the fraction of changed labels falls below a tolerance.

For each anchor $\mathbf{x}_i$ with current class identity $c_i = \tilde{y}_i^{(t)}$, we create two views $\widetilde{\mathbf{x}}_i^{(1)}$ and $\widetilde{\mathbf{x}}_i^{(2)}$ by replacing a small subset of the anchor's features. Let d denote the number of features (coordinates) of $\mathbf{x}_i$, and let $r \in (0,1)$ be the replacement rate. In each view, we randomly select exactly $\lceil rd \rceil$ features of the anchor to replace. Denote by $\mathcal{I}^{(t)} \subset \{1,\dots,d\}$ the index set chosen for view $t \in \{1,2\}$; the two sets are sampled independently (preferably with minimal overlap). For every index $j \in \mathcal{I}^{(t)}$, the new value is drawn from the empirical distribution of the same feature among flows currently labeled c_i, while all other features are kept equal to the anchor:

$$\begin{aligned} \widetilde{\mathbf{x}}_i^{(t)}[j] &\sim P(\mathbf{x}[j] \,|\, \tilde{y} = c_i)\,, \quad && j \in \mathcal{I}^{(t)}, \\ \widetilde{\mathbf{x}}_i^{(t)}[k] &= \mathbf{x}_i[k], && k \notin \mathcal{I}^{(t)}. \end{aligned} \tag{10}$$

This class-conditioned replacement (TabCL principle [36]) keeps views semantically close to the anchor $\mathbf{x}_i$ while inducing invariances within the class manifold.

After replacement, we update each view to satisfy simple algebraic identities among continuous features, using the residual set $\mathcal{G}$ defined in the AE subsection. In practice, we adjust the augmented vector only as much as needed to restore identities, keeping the view close to the original replacement outcome while making the tuple plausible in terms of inter-feature consistency. Let $\breve{\mathbf{x}}_i^{(t)}$ denote the adjusted view:

$$\begin{aligned} \breve{\mathbf{x}}_i^{(t)} &= \Pi_{\mathcal{G}}\big(\widetilde{\mathbf{x}}_i^{(t)}\big), \\ &= \arg\min_{\mathbf{u}} \ \big\|\mathbf{u} - \widetilde{\mathbf{x}}_i^{(t)}\big\|_2^2 \\ &\quad \text{s.t.} \ \ g_m\big(\mathbf{u}_{\text{cont}}\big) = 0, \ \forall g_m \in \mathcal{G}. \end{aligned}$$

Here, $\mathbf{u}$ is the optimization variable (the adjusted vector we solve for), $\mathbf{u}_{\text{cont}}$ is its continuous slice, $\Pi_{\mathcal{G}}(.)$ denotes the projection operator that returns the nearest vector that satisfies all constraints in $\mathcal{G}$, and $\arg\min$ selects the point that minimizes the squared distance to the replaced view. This constraint-preserving update is simple (closed-form or single-pass updates for ratio/product/sum identities) and ensures that positives are plausible flow tuples. Although applied at the view level,

the induced gradients during pretraining bias the encoder's representation toward respecting such identities, which we later reuse for supervised fine-tuning.

Dual-Head Projections for Heterogeneous Features

After creating two augmented views by class-conditioned replacement and constraint-preserving update in the pretraining phase, we adopt NT-Xent loss 2 to promote similarity between augmented views. For each mini-batch containing N anchors, we form two augmented views per anchor and each view is encoded by the shared backbone and then mapped by two independent projection heads tailored to the feature types: we compute the embedding $\mathbf{z}_i^{(t)} = \mathcal{E}(\breve{\mathbf{x}}_i^{(t)})$, feed it to a continuous-focused projector $\mathcal{P}_{\text{cont}}$ to obtain $\mathbf{v}_{\text{cont},i}^{(t)} = \mathcal{P}_{\text{cont}}(\mathbf{z}_i^{(t)})$, and to a categorical-focused projector $\mathcal{P}_{\text{cat}}$ to obtain $\mathbf{v}_{\text{cat},i}^{(t)} = \mathcal{P}_{\text{cat}}(\mathbf{z}_i^{(t)})$ for $t \in \{1, 2\}$ and $i = 1, \ldots, N$. Each anchor, therefore, appears twice (once per view) as the "query," and for each query we use its counterpart view as the positive and all other in-batch projected views from the same head as negatives. To preserve symmetry, we evaluate NT-Xent in both directions (view 1 as anchor with view 2 as positive, and vice versa) and average across the resulting $2N$ anchoring per head. The head-specific losses are thus

$$\mathcal{L}_{\text{cont}} = \frac{1}{2N} \sum_{i=1}^{N} \Big[\mathcal{L}_{\text{NT-Xent}}\big(\mathbf{v}_{\text{cont},i}^{(1)}, \mathbf{v}_{\text{cont},i}^{(2)}, \tau_{\text{cont}}\big) + \mathcal{L}_{\text{NT-Xent}}\big(\mathbf{v}_{\text{cont},i}^{(2)}, \mathbf{v}_{\text{cont},i}^{(1)}, \tau_{\text{cont}}\big) \Big]. \tag{11}$$

$$\mathcal{L}_{\text{cat}} = \frac{1}{2N} \sum_{i=1}^{N} \Big[\mathcal{L}_{\text{NT-Xent}}\big(\mathbf{v}_{\text{cat},i}^{(1)}, \mathbf{v}_{\text{cat},i}^{(2)}, \tau_{\text{cat}}\big) + \mathcal{L}_{\text{NT-Xent}}\big(\mathbf{v}_{\text{cat},i}^{(2)}, \mathbf{v}_{\text{cat},i}^{(1)}, \tau_{\text{cat}}\big) \Big]. \tag{12}$$

The overall contrastive objective combines the two heads with a convex weight:

$$\mathcal{L}_{\text{TabCL}} \;=\; \lambda\, \mathcal{L}_{\text{cont}} \;+\; (1-\lambda)\, \mathcal{L}_{\text{cat}}, \qquad \lambda \in [0, 1]. \tag{13}$$

In our framework, instead of a single projection head with a single temperature shared across all features, we employ dual-head projections with head-specific temperatures. This design acknowledges that network-flow tables are heterogeneous features. Using distinct temperatures is beneficial because categorical similarities are typically sharper (best handled with a smaller τ_{cat}), while continuous similarities are more variable (more stable with a larger τ_{cont}). The shared encoder thus learns

representations that are simultaneously robust for both feature types, while each head specializes its geometry (via temperature) to the statistics of its slice.

Fine-Tuning, Pseudo-Labeling, and Hyperparameters

After contrastive pretraining, we discard $\mathcal{P}_{\text{cont}}$ and $\mathcal{P}_{\text{cat}}$, attach a classifier $\mathcal{C}(\,\cdot\,;\theta_{\mathcal{C}})$ to $\mathcal{E}$, and fine-tune on D_s using the freeze–unfreeze schedule and cross-entropy (cf. Eq. (9)). We then apply $\mathcal{C}\circ\mathcal{E}$ to all $\mathbf{x} \in D_l$ to obtain final pseudo-labels. The hyperparameters such as temperatures $(\tau_{\text{cat}}, \tau_{\text{cont}})$, the mixing coefficient λ, the replacement rate r, and the refresh schedule are selected on D_s via stratified cross-validation to maximize macro-F1 after fine-tuning. Algorithm 2 details the expanded procedure.

Algorithm 2 TabCL with Class-Conditioned Replacement, Constraint Projection, and Dual-Head Objective

Require: Unlabeled D_l, labeled D_s; replacement rate r; refresh interval; temperatures $\tau_{\text{cont}}, \tau_{\text{cat}}$; mixing λ

Bootstrap: train a light classifier on D_s; get initial $\{\tilde{y}_i^{(0)}\}$ for D_l
Initialize encoder $\mathcal{E}$ and projectors $\mathcal{P}_{\text{cont}}, \mathcal{P}_{\text{cat}}$
for epoch $= 1, 2, \ldots$ **do**
    if refresh interval reached **then**
        Freeze $\mathcal{E}$; train a linear probe on $\{(\mathcal{E}(\mathbf{x}), y)\}_{D_s}$; reclassify $\{\mathcal{E}(\mathbf{x})\}_{D_l}$ to obtain $\{\tilde{y}_i^{(t)}\}$; stop refreshing if label-change fraction $<$ tolerance or max rounds reached.
    end if
    for mini-batch $\{\mathbf{x}_i\}_{i=1}^{N} \subset D_l$ **do**
        For each i: set $c_i \leftarrow \tilde{y}_i^{(t)}$; sample index sets $\mathcal{I}^{(1)}, \mathcal{I}^{(2)}$ of size $\lceil rd \rceil$
        For $t \in \{1, 2\}$: replace $\widetilde{\mathbf{x}}_i^{(t)}[j] \sim P(\mathbf{x}[j] \mid \tilde{y} = c_i)$ for $j \in \mathcal{I}^{(t)}$, copy others from $\mathbf{x}_i$; then project to constraints $\breve{\mathbf{x}}_i^{(t)} \leftarrow \Pi_{\mathcal{G}}(\widetilde{\mathbf{x}}_i^{(t)})$ as in (11).
        Encode/project: $\mathbf{z}_i^{(t)} \leftarrow \mathcal{E}(\breve{\mathbf{x}}_i^{(t)})$; $\mathbf{v}_{\text{cont},i}^{(t)} \leftarrow \mathcal{P}_{\text{cont}}(\mathbf{z}_i^{(t)})$; $\mathbf{v}_{\text{cat},i}^{(t)} \leftarrow \mathcal{P}_{\text{cat}}(\mathbf{z}_i^{(t)})$
    end for
    Compute $\mathcal{L}_{\text{cont}}$ and $\mathcal{L}_{\text{cat}}$ via (11), (12)
    Compute $\mathcal{L}_{\text{TabCL}} = \lambda\mathcal{L}_{\text{cont}} + (1-\lambda)\mathcal{L}_{\text{cat}}$ (13); update parameters
end for
Discard projectors; attach $\mathcal{C}$ to $\mathcal{E}$; fine-tune on D_s with (9); pseudo-label D_l

Ensure: Pseudo-labeled D_l from TabCL

3.3.4 Confidence–Margin Fusion of AE and TabCL

Either SSL branch (AE or TabCL) can independently be used to produce a pseudo-labeled pool for the subsequent CL stage. When both are available, we adopt a simple voting rule that consolidates their predictions at negligible overhead. For each flow $\mathbf{x}$, let

$$\widehat{c}^{\,\mathrm{AE}}(\mathbf{x}) = \arg\max_{k\in\{1,\dots,K\}} p_k^{\mathrm{AE}}(\mathbf{x}), \tag{14}$$

$$\widehat{c}^{\,\mathrm{TabCL}}(\mathbf{x}) = \arg\max_{k\in\{1,\dots,K\}} p_k^{\mathrm{TabCL}}(\mathbf{x}), \tag{15}$$

be the top-1 classes predicted by the AE and TabCL classifiers, where $p^{\mathrm{AE}}(\mathbf{x}), p^{\mathrm{TabCL}}(\mathbf{x}) \in [0,1]^K$ are their respective softmax probability vectors. Define the *confidence* of each branch as its top-1 probability,

$$s^{\mathrm{AE}}(\mathbf{x}) \;=\; \max_k p_k^{\mathrm{AE}}(\mathbf{x}), \qquad s^{\mathrm{TabCL}}(\mathbf{x}) \;=\; \max_k p_k^{\mathrm{TabCL}}(\mathbf{x}), \tag{16}$$

and the *margin* (gap between the highest and the second-highest probabilities),

$$m^{\mathrm{AE}}(\mathbf{x}) \;= s^{\mathrm{AE}}(\mathbf{x}) \;-\; \max_{k\neq\widehat{c}^{\,\mathrm{AE}}} p_k^{\mathrm{AE}}(\mathbf{x}), \tag{17}$$

$$m^{\mathrm{TabCL}}(\mathbf{x}) \;= s^{\mathrm{TabCL}}(\mathbf{x}) \;-\; \max_{k\neq\widehat{c}^{\,\mathrm{TabCL}}} p_k^{\mathrm{TabCL}}(\mathbf{x}). \tag{18}$$

The fused pseudo-label $\widehat{c}(\mathbf{x})$ is decided by agreement, then by confidence, then by margin:

$$\widehat{c}(\mathbf{x}) \;=\; \begin{cases} \widehat{c}^{\,\mathrm{AE}}(\mathbf{x}) & \text{if } \widehat{c}^{\,\mathrm{AE}}(\mathbf{x}) = \widehat{c}^{\,\mathrm{TabCL}}(\mathbf{x}), \\ \widehat{c}^{\,\mathrm{AE}}(\mathbf{x}) & \text{if } s^{\mathrm{AE}}(\mathbf{x}) > s^{\mathrm{TabCL}}(\mathbf{x}), \\ \widehat{c}^{\,\mathrm{TabCL}}(\mathbf{x}) & \text{if } s^{\mathrm{TabCL}}(\mathbf{x}) > s^{\mathrm{AE}}(\mathbf{x}), \\ \widehat{c}^{\,\mathrm{AE}}(\mathbf{x}) & \text{if } s^{\mathrm{AE}}(\mathbf{x}) = s^{\mathrm{TabCL}}(\mathbf{x}) \\ & \text{and } m^{\mathrm{AE}}(\mathbf{x}) \geq m^{\mathrm{TabCL}}(\mathbf{x}) \\ \widehat{c}^{\,\mathrm{TabCL}}(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{19}$$

This rule favors the model with the higher top-1 confidence; if confidences tie exactly, the larger margin (sharper decision) prevails. In practice, exact ties are exceedingly rare; the last line provides a deterministic fallback. The resulting single pseudo-labeled set is then passed to the downstream task. The process of fusion is detailed in Algorithm 3.

Algorithm 3 Confidence–Margin Voting Fusion of AE and TabCL

Require: Trained AE and TabCL classifiers; unlabeled D_l
Ensure: Fused pseudo-labels on D_l

for each $\mathbf{x} \in D_l$ **do**
  $\mathbf{p}^{\text{AE}} \leftarrow$ softmax from AE; $\mathbf{p}^{\text{TabCL}} \leftarrow$ softmax from TabCL
  $\widehat{c}^{\text{AE}} \leftarrow \arg\max_k \mathbf{p}^{\text{AE}}[k]$; $\widehat{c}^{\text{TabCL}} \leftarrow \arg\max_k \mathbf{p}^{\text{TabCL}}[k]$ (14)
  if $\widehat{c}^{\text{AE}} = \widehat{c}^{\text{TabCL}}$ **then**
    $\widehat{c} \leftarrow \widehat{c}^{\text{AE}}$
  else
    $s^{\text{AE}} \leftarrow \max_k \mathbf{p}^{\text{AE}}[k]$; $s^{\text{TabCL}} \leftarrow \max_k \mathbf{p}^{\text{TabCL}}[k]$ (16)
    $m^{\text{AE}} \leftarrow s^{\text{AE}} - \max_{k \neq \widehat{c}^{\text{AE}}} \mathbf{p}^{\text{AE}}[k]$; $m^{\text{TabCL}} \leftarrow s^{\text{TabCL}} - \max_{k \neq \widehat{c}^{\text{TabCL}}} \mathbf{p}^{\text{TabCL}}[k]$ (17)
    Decide $\widehat{c}$ by rule (19): prefer higher confidence; if tied, prefer larger margin; fall back deterministically otherwise.
  end if
  Assign $\widehat{c}$ to $\mathbf{x}$
end for

Fusion often improves corner cases where the two branches disagree on difficult classes, with minimal changes to the pipeline. The cost is additional SSL pretraining (both branches), which we report explicitly in the efficiency section, together with parameter counts and epochs for each path.

3.4 Confident Learning and Noise-Aware Final Training

The pseudo-labeled pool $D_l = \{(\mathbf{x}_i, \tilde{y}_i)\}_{i=1}^{|D_l|}$ produced by SSL, as detailed in the preceding subsections, inevitably contains noise. Training directly on these labels degrades performance. We aim to reduce the impact of noisy pseudo-labels before training the final classifier. We therefore adopt and extend CL [37] into a pipeline that down-weights noisy pseudo-labels and trains the final classifier in five stages: First, following standard CL practice, we compute out-of-sample prediction probabilities and define each sample's self-confidence. Second, we introduce class-adaptive, quantile-based confidence thresholds with robust dispersion (MAD) to reflect per-class difficulty.

Third, we convert confidence into smooth, calibration-aware per-sample logistic weights that attenuate uncertain pseudo-labels while retaining all data. Fourth, leveraging the CL confident joint, we perform a class-wise balanced retention calibration that aligns each class's retained mass with its estimated clean fraction, mitigating imbalance under label noise. Finally, we train the final classifier $\mathcal{C}_{\text{final}}$ using weighted symmetric cross-entropy on the full pseudo-labeled set. Importantly, we do not remove samples; rather, we reduce the contribution of uncertain ones and calibrate class masses to stabilize learning.

3.4.1 Prediction Probabilities and Self-Confidence

We first obtain out-of-sample predicted probabilities $\mathbf{p}_i$ for each sample $\mathbf{x}_i$ in D_l using F-fold cross-validation with a simple (a shallow MLP) classifier, for each sample $\mathbf{x}_i$, use the model trained on the $F-1$ folds that did not include $\mathbf{x}_i$. Denoting the parameters of this model by $\theta^{(-f(i))}$, we calculate the K-way probability vector:

$$\begin{aligned} \mathbf{p}_i \;=\; \big[\, P\big(y{=}1 \mid \mathbf{x}_i; \theta^{(-f(i))}\big),\, \ldots,\, P\big(y{=}K \mid \mathbf{x}_i; \theta^{(-f(i))}\big) \,\big], \\ \mathbf{p}_i \in [0,1]^K, \qquad \sum_{k=1}^{K} \mathbf{p}_i[k] = 1. \end{aligned} \tag{20}$$

In other words, $\mathbf{p}_i$ is $P(y \mid \mathbf{x}_i)$ evaluated for all classes. The top-1 class of this out-of-sample model is $\hat{y}_i^\star = \arg\max_k \mathbf{p}_i[k]$, which may or may not equal the SSL pseudo-label $\tilde{y}_i$. In CL, the quantity of interest is self-confidence s_i, defined as the probability that the out-of-sample model assigns to the observed (pseudo) label of the sample:

$$s_i \;=\; \mathbf{p}_i[\tilde{y}_i] \;\in\; [0,1]. \tag{21}$$

Thus, $\mathbf{p}_i$ is a K-dimensional probability vector ($P(y \mid \mathbf{x}_i)$ across all classes), while s_i is a single scalar entry of that vector corresponding to $\tilde{y}_i$. These out-of-sample predictions and the resulting self-confidences are used exclusively to vet pseudo-labels via CL; they do not provide gradients to $\mathcal{C}_{\text{final}}$.

3.4.2 Per-Class, Quantile-Calibrated Thresholds

For each observed class $j \in \{1, \ldots, K\}$, collect the self-confidence scores of samples whose pseudo-label equals j: $\mathcal{S}_j = \{\, s_i : \tilde{y}_i = j \,\}$, and $n_j = |\mathcal{S}_j|$. Let the elements of $\mathcal{S}_j$ in non-decreasing order be $s_{j,(1)} \leq s_{j,(2)} \leq \cdots \leq s_{j,(n_j)}$ (order statistics). For a quantile level $q \in (0, 1]$, define the per-class q-quantile threshold by the standard order-statistic rule:

$$\begin{aligned} t_j^{(q)} &= \text{Quantile}_q(\mathcal{S}_j) = s_{j,\,(k_j(q))}, \\ k_j(q) &= \lceil q\, n_j \rceil\,, \qquad q \in (0, 1]. \end{aligned} \tag{22}$$

(Equivalently, $t_j^{(q)}$ is the smallest value for which at least a fraction q of the scores in $\mathcal{S}_j$ are $\leq t_j^{(q)}$; linear interpolation can be used if a specific quantile convention is desired.)

We treat q as a hyperparameter shared across classes and select it by stratified cross-validation on the small labeled set D_s. This procedure ties the threshold choice directly to downstream performance while preserving the per-class adaptivity of $t_j^{(q)}$.

Canonical CL [37] uses the per-class mean self-confidence as a threshold $t_j^{(\text{mean})} = \bar{S}_j$. In contrast, we set the class threshold to a quantile of $\mathcal{S}_j$, which offers two advantages for network traffic flows: (i) it adapts to class-dependent score distributions that are often skewed due to heterogeneous or encrypted traffic patterns, making the cutoff robust to outliers; and (ii) it provides an explicit strictness control via q: higher q yields a stricter threshold (fewer samples lie above $t_j^{(q)}$ and will later receive large weights), while lower q is more permissive.

3.4.3 Calibration-Aware Per-Sample Weights

To avoid hard pruning (outright removal of suspected-noisy samples) and to stabilize training, we define a smooth weight for each sample that depends on its distance to the class threshold and on the dispersion of confidences in that class. For each observed class, we define the class median by:

$$\tilde{m}_j = \text{median}\big(\mathcal{S}_j\big). \tag{23}$$

We measure dispersion using the *Median Absolute Deviation (MAD)*:

$$\mathrm{MAD}_j \;=\; \mathrm{median}\big\{\, |s - \tilde{m}_j| : s \in \mathcal{S}_j \,\big\}. \tag{24}$$

We set the classwise scale to σ_j and clip it to avoid division by zero:

$$\sigma_j \;\leftarrow\; \max(\mathrm{MAD}_j, \varepsilon), \quad \varepsilon > 0. \tag{25}$$

Using MAD makes the scale estimate robust to outliers and skewed score distributions common in traffic flows. Given the per-class quantile threshold $t_j^{(q)}$ from (22), define for sample i with $\tilde{y}_i = j$:

$$z_i \;=\; \frac{s_i - t_{\tilde{y}_i}^{(q)}}{\gamma\, \sigma_{\tilde{y}_i}}, \tag{26}$$

where $\gamma > 0$ controls the slope around the threshold. We map z_i to $(0, 1)$ with the logistic function that is:

$$\mathrm{sigm}(z_i) \;=\; \frac{1}{1 + e^{-z_i}}. \tag{27}$$

With a minimum weight $w_{\min} \in (0, 1)$, the per-sample training weight is:

$$w_i \;=\; w_{\min} \;+\; (1 - w_{\min}) \cdot \mathrm{sigm}(z_i) \;\;\in\; [\, w_{\min},\, 1\,]. \tag{28}$$

All hyperparameters in (25)–(28) (e.g., $w_{\min}$, γ, ε) are selected via stratified cross-validation on the small labeled set D_s.

The weighting mechanism in (28) behaves in a stable and interpretable manner. By construction $w_i \in [w_{\min}, 1]$; When $s_i \ll t_{\tilde{y}_i}^{(q)}$, the pseudo-label of the flow is likely noisy, the standardized margin $z_i \to -\infty$ and thus $w_i \to w_{\min}$. Conversely, when $s_i \gg t_{\tilde{y}_i}^{(q)}$, the pseudo-label is likely correct, $z_i \to +\infty$ and $w_i \to 1$. At the threshold, if $s_i = t_{\tilde{y}_i}^{(q)}$ then $z_i = 0$ and $w_i = w_{\min} + \frac{1}{2}(1 - w_{\min})$, i.e., the exact midpoint between the minimum and maximum weights.

3.4.4 Confident Joint and Class-Wise Balanced Retention

In CL, the matrix $\widehat{Q} \in R^{K\times K}$ summarizes the estimated joint between the latent true class (rows) and the pseudo-label (columns). It is computed directly from out-of-sample probability vectors $\mathbf{p}_i$ using soft counts. Specifically,

$$\widehat{Q}_{y=k,\,\tilde{y}=j} \;=\; \sum_{i:\,\tilde{y}_i=j} \mathbf{p}_i[k], \qquad k,j \in \{1,\ldots,K\}, \tag{29}$$

where the row index "$y = k$" denotes the latent class inferred from $\mathbf{p}_i$. The diagonal entry $\widehat{Q}_{j,j}$ estimates the clean mass for class j, while the column sum $\sum_k \widehat{Q}_{k,j}$ is the total mass observed as j. The corresponding estimated clean fraction for observed class j is:

$$\rho_j \;=\; \frac{\widehat{Q}_{j,j}}{\sum_k \widehat{Q}_{k,j}} \;\in\; [0,1]. \tag{30}$$

To mitigate minority erosion under class imbalance, a Balanced Retention Constraint (BRC) is applied after computing per-sample weights w_i. Define $n_j = \{i : \tilde{y}_i = j\}$, the target effective mass $T_j = \rho_j\, N_j$, and the current mass $M_j = \sum_{i:\,\tilde{y}_i=j} w_i$. The classwise scaling factor is:

$$a_j \;=\; \frac{T_j}{\max(\epsilon,\, M_j)}, \tag{31}$$

where $\epsilon > 0$ provides numerical stability. The final per-sample weight used in training is then:

$$w_i' \;=\; \mathrm{clip}\big(a_j\, w_i,\, w_{\min},\, 1\big), \tag{32}$$

This soft constraint yields $\sum_{i:\,\tilde{y}_i=j} w_i' \approx T_j$ (within a small tolerance), aligning the retained mass of each observed class with its clean fraction estimated from (29)–(30).

If $M_j < T_j$, then $a_j > 1$ and all weights in class j are uniformly lifted (subject to clipping), preventing under-retention of harder/minority classes; if $M_j > T_j$, then $a_j < 1$ and the class is gently down-scaled to avoid over-retention. Because $w_i \in [w_{\min}, 1]$, the effective mass per class lies in $[w_{\min} n_j,\, n_j]$; the BRC moves it toward $T_j = \rho_j N_j$, thereby preserving minority classes in proportion to their estimated cleanliness. Equation (29) follows the standard confident joint

construction in CL, whereas the scaling rule in (31) and (32) constitutes the proposed extension that operationalizes class-aware retention during training.

3.4.5 Final Training with Weighted SCE

The downstream classifier $\mathcal{C}_{\text{final}}$ is trained on the pseudo-labeled dataset $D_l = \{(\mathbf{x}_i, \tilde{y}_i)\}$ using the Symmetric Cross-Entropy (SCE) loss, weighted by the final per-sample weight w_i' from (32). For a sample $(\mathbf{x}_i, \tilde{y}_i)$ with predicted probability vector $\mathbf{p}_i = \mathcal{C}_{\text{final}}(\mathbf{x}_i) \in [0,1]^K$, the loss is:

$$\mathcal{L}_{\text{SCE}}^{(i)} \;=\; w_i' \left[\, \alpha\, \text{CE}(\mathbf{p}_i, \tilde{y}_i) \;+\; \beta\, \text{RCE}(\tilde{y}_i, \mathbf{p}_i) \,\right]. \tag{33}$$

The coefficients $\alpha, \beta > 0$ are hyperparameters and are selected by stratified cross-validation on the small labeled set D_s. We adopt SCE because it combines the standard cross-entropy (encouraging correct predictions) with a reverse cross-entropy term that penalizes overconfident mistakes, improving robustness under residual label noise in pseudo-labeled data. Concretely,

$$\hat{y}_{i,k} \;=\; \begin{cases} 1-\epsilon, & k = \tilde{y}_i, \\ \dfrac{\epsilon}{K-1}, & k \neq \tilde{y}_i, \end{cases} \qquad \epsilon \in (0,1). \tag{34}$$

$$\begin{aligned} \text{CE}(\mathbf{p}_i, \tilde{y}_i) \;&=\; -\sum_{k=1}^{K} \hat{y}_{i,k}\, \log \mathbf{p}_i[k], \\ \text{RCE}(\tilde{y}_i, \mathbf{p}_i) \;&=\; -\sum_{k=1}^{K} \mathbf{p}_i[k]\; \log \hat{y}_{i,k}. \end{aligned} \tag{35}$$

Then, parameters of $\mathcal{C}_{\text{final}}$ are updated by gradient descent:

$$\theta_{\text{final}} \;\leftarrow\; \theta_{\text{final}} \;-\; \eta\, \nabla_{\theta_{\text{final}}} \mathcal{L}_{\text{SCE}}. \tag{36}$$

According to the mathematical formulations and procedural details outlined above, our innovations make this foundation more suitable for the unique challenges of network traffic. Unlike original CL, which relies on mean-based thresholds that can be sensitive to skewed confidence distributions common in imbalanced and heterogeneous traffic data, our quantile-calibrated thresholds

provide a more robust cutoff by adapting to class-specific variation and outliers; this avoids overly strict filtering for hard classes and overly lax filtering for easy ones. Similarly, while the canonical method often uses hard pruning that risks losing valuable samples, our logistic weights enable nuanced attenuation, retaining partial contributions from uncertain instances to preserve information and maintain dataset richness. Furthermore, the balanced retention mechanism addresses class imbalance by scaling masses according to clean estimates, a step not present in the original that helps preserve minority flows. Together, these enhancements make our module more effective for traffic classification, where class skew and variable noise are prevalent. Beyond this intuition, our ablations and experiments across different datasets (section 3.5) confirm that this traffic-adapted CL outperforms the original approach in terms of overall classification performance. The complete CL-driven noise mitigation and downstream training procedure is summarized in Algorithm 4.

Algorithm 4 Traffic-Adapted CL with Quantile Thresholds, Logistic Weights, BRC, and Weighted SCE

Require: Pseudo-labeled $D_l = \{(\mathbf{x}_i, \tilde{y}_i)\}$; small labeled D_s; folds F; grids $\mathcal{Q}$ (quantile q), $\mathcal{W}$ ($w_{\min}$), Γ (slope γ)

Ensure: Trained final classifier $\mathcal{C}_{\text{final}}$

Out-of-sample probabilities: obtain $\mathbf{p}_i$ via F-fold CV (20); set $s_i \leftarrow \mathbf{p}_i[\tilde{y}_i]$ (21)
Thresholds/weights per class:
for $q \in \mathcal{Q}$, $w_{\min} \in \mathcal{W}$, $\gamma \in \Gamma$ **do**
    for each class j **do**
        Compute $t_j^{(q)}$ (22), $\tilde{m}_j$ (23), MAD_j (24), σ_j (25)
    end for
    for each i **do**
        $z_i \leftarrow (s_i - t_{\tilde{y}_i}^{(q)})/(\gamma\, \sigma_{\tilde{y}_i})$ (26)
        $w_i \leftarrow w_{\min} + (1 - w_{\min}) \cdot \mathrm{sigm}(z_i)$ (28)
    end for
    Build confident joint $\widehat{Q}$ (29); get ρ_j (30); compute a_j (31)
    Final weights: $w_i' \leftarrow \mathrm{clip}(a_j\, w_i,\, w_{\min},\, 1)$ (32) for $\tilde{y}_i = j$
    Train $\mathcal{C}_{\text{final}}$ on D_l with weighted SCE (33)
end for
Deployment: fix $(q, w_{\min}, \gamma)$ according to the chosen configuration; recompute $\{w_i'\}$ on the full pseudo-labeled pool and train $\mathcal{C}_{\text{final}}$ on D_l.

3.5 Experimental Evaluation and Results

This section presents the evaluation of the framework using standard metrics (accuracy, precision, recall, and macro-F1). The analysis covers pseudo-label quality from AE/TabCL (and their fusion), the effect of CL and its components, the resulting classifier performance, computational profile, and comparisons against contemporary baselines.

3.5.1 Experimental Setup and Datasets

Our assessment relies on two datasets: a self-generated dataset used as unlabeled input for SSL and a compact labeled dataset derived from ISCX VPN-nonVPN [14]. We include the self-generated data to emulate a realistic deployment pipeline, starting from raw packet captures and ending with flow-level tabular features, while reflecting practical label scarcity and class imbalance. To extend coverage and examine generalization to contemporary encrypted traffic, we additionally evaluate on UCDavis–QUIC [39]. Below, we provide a detailed description of each dataset, including its creation process, key characteristics, and its function within the overall evaluation.

Self-Generated Unlabeled Dataset: The unlabeled dataset was constructed using Global Network Simulator 3 (GNS3) to emulate a realistic network scenario. A simulated Cisco router with NetFlow enabled served as the exporter of IP flow records, which were transmitted to a VMware-hosted virtual server operating as the NetFlow collector. Traffic was generated on a Windows 10 virtual machine running ten applications selected to cover diverse categories: VPN traffic (VPN_Vimeo, Tor_YouTube), time-sensitive traffic (Skype_VideoCall, Facebook_Audio), file-transfer traffic (FTPS-Upload, SFTP_Upload, SCP_Download), and video/voice traffic (YouTube, Skype_Chat, Email). This mix was designed to reflect common usage patterns. Figure 3.4 presents the overall architecture and the data-generation workflow.

Processing of the captured NetFlow data was performed on the collector by a custom Python script that executed the feature pipeline. The script extracted core statistics such as packet count, octet count, and throughput, and it computed time-related descriptors including duration and inter-arrival times. In an additional generation step, the raw NetFlow packets were aggregated to obtain minimum, maximum, mean, and standard deviation for selected features. The final dataset is stored

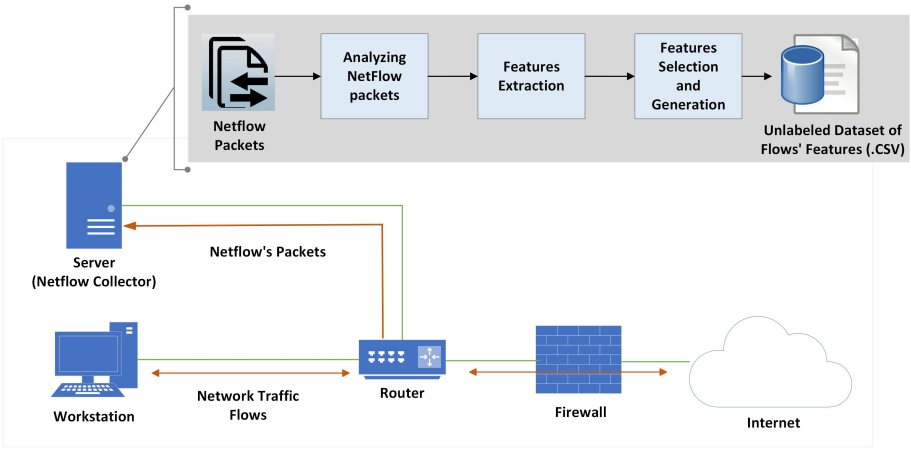


Figure 3.4: Network architecture and Flowchart of the unlabeled dataset generation.

as a Comma-Separated Values (CSV) file and comprises 21 features and 8,684 flows. The distribution is skewed in a way that mirrors operational traffic: YouTube contributes 41.9% of flows, while SFTP_Upload and SCP_Download each remain below 4.5%. This imbalance is characteristic of NTC and is mitigated by class-aware thresholds together with balanced retention in the CL stage. The feature distributions are consistent with networking constraints and display heavy tails: the median Mean Packet Length is 571 B (95th percentile = 1,406 B, below Ethernet MTU), and the median Mean IAT is 6.9 ms (95th percentile = 0.59 s).

ISCX VPN-nonVPN as a Small Labeled Dataset: The smaller labeled dataset was prepared from the ISCX VPN-nonVPN corpus [14], a widely used public resource for NTC studies. The source material is provided at the packet level as PCAP files that record detailed network activity from varied scenarios. It includes a mixture of VPN and non-VPN traffic and covers both encrypted and unencrypted flows across different applications and network states, providing a realistic test bed for traffic classification. To adapt it to our setting, custom Python scripts converted the packet traces to flow-level by analyzing packet sequences and grouping them into flows using attributes such as source and destination IP addresses and port numbers. From these records, 538 complete flows were selected to reduce labeling effort while preserving alignment with the ten application categories used in the larger dataset. This collection functions as the labeled set for model fine-tuning even though it is much smaller than the self-generated data. The resulting class distribution is intentionally imbalanced to reflect real traffic mixes; Table 3.2 reports the per-class distribution

in both datasets.

Table 3.2: The self-generated and ISCX VPN-nonVPN class distribution.

Class	**Number of Flows**		**Share (%)**
	Self generated Dataset	ISCX Dataset	
YouTube	3764	84	41.85
Skype_VideoCall	438	63	5.44
Skype_Chat	956	41	10.80
FTPS_Upload	638	53	7.48
SFTP_Upload	321	20	3.69
SCP_Download	387	22	4.44
Facebook_Audio	497	59	6.04
Email	904	98	10.86
Tor_YouTube	359	44	4.38
VPN_Vimeo	416	53	5.01
Total	**8684**	**538**	**100.00**

UCDavis–QUIC Dataset: Evaluation is also conducted on the UCDavis–QUIC dataset, which focuses exclusively on QUIC traffic collected from five Google services: Docs, Drive, Music, Search, and YouTube. The dataset is released as packet captures, and our processing converts PCAP to CSV and aggregates records by the five-tuple to produce flow-level statistics. According to the original publication [39], each flow is described by twenty-four numerical features, and a separate categorical variable records the direction of the flow as forward, backward, or bidirectional. The dataset totals 6,439 flows. To match the label-scarce assumption, 5% of the flows per class are treated as labeled, and the remaining 95% are treated as unlabeled. Table 3.3 presents the per-class counts and the stratified labeled and unlabeled partition.

Table 3.3: UCDavis–QUIC class distribution.

Class	**Total Flows**	**Labeled**	**Unlabeled**	**Share (%)**
Google Docs	1221	61	1160	18.96
Google Drive	1634	82	1552	25.38
Google Music	592	30	562	9.19
Google Search	1915	96	1819	29.74
YouTube	1077	54	1023	16.73
Total	**6439**	**323**	**6116**	**100.00**

We conducted our experiments on a workstation running Ubuntu 22.04.4 LTS, Python 3.12.11, and PyTorch 2.8.0+cu126, equipped with an Intel Xeon CPU @ 2.00 GHz, 12.7 GB RAM, and a

single NVIDIA Tesla T4 (15.0 GB) GPU.

3.5.2 Evaluation Metrics

In the evaluation, model performance is reported using standard classification metrics. Throughout, TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

- **Accuracy**: The proportion of correctly classified samples. This metric provides an overall measure of the model's correctness across all classes, defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{37}$$

- **Recall**: The ratio of correctly identified positive samples. Recall indicates the model's ability to detect all relevant instances of a class, given by:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{38}$$

- **Precision**: The proportion of correctly predicted positive samples among all samples predicted as positive, calculated as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{39}$$

- **F1-Score**: The harmonic mean of precision and recall. The F1-score balances precision and recall, offering a single metric that reflects the performance of the model, computed as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{40}$$

Together, these metrics provide a concise and robust view of the model's classification performance.

3.5.3 Results on Self-Generated and ISCX Datasets

The analysis in this subsection covers performance on the Self-generated and ISCX datasets, focusing on pseudo-label quality produced by SSL and on the final accuracy achieved once CL is applied to reduce noise in those labels. Throughout training, the Self-generated dataset remained unlabeled, and its ground-truth labels were accessed only for evaluation. This setup allows SSL to learn structure from abundant unlabeled data, while the small labeled dataset is used for fine-tuning. This design simultaneously mitigates label scarcity while maximizing utility from both sources.

About the hyperparameter selection, we tuned each module with a stratified 5-fold cross-validation (CV) on the small labeled ISCX split; we repeated training with 3 random seeds and reported Macro-F1 as mean ± 95% CI over the resulting 15 scores (5 folds × 3 seeds). We then selected the best settings per module by Macro-F1. Tables 3.4–3.6 present representative configurations from the full grid used to select hyperparameters for AE, TabCL, CL, and the final classifier.

Table 3.4: Representative grid search results for AE's hyperparameters.

Latent dim	Depth	Hidden width per layer	Dropout	LR	Weight decay	Batch	Epochs	ϕ (constraint)	Macro-F1 (%)
128	3	(256, 128, 64)	0.10	5×10^{-4}	1×10^{-5}	128	100	0.5	**93.8 ± 0.6**
64	2	(256, 128)	0.20	5×10^{-4}	1×10^{-5}	256	100	0.1	92.6 ± 0.6
32	3	(256, 128, 64)	0.00	1×10^{-3}	1×10^{-6}	128	50	1.0	91.2 ± 0.9
64	2	(128, 64)	0.30	1×10^{-3}	1×10^{-4}	64	50	0.0	89.7 ± 0.9
128	3	(256, 128, 64)	0.10	2×10^{-4}	1×10^{-5}	256	80	1.0	92.9 ± 0.5
64	3	(256, 128, 64)	0.20	5×10^{-4}	1×10^{-4}	64	50	2.0	90.1 ± 0.8

Table 3.5: Representative grid search results for TabCL's hyperparameters.

Depth	Hidden widths	LR	Weight decay	Batch	Epochs	τ_{cont}	τ_{cat}	r	λ	Proj dim	Refresh	Macro-F1 (%)
3	(256, 128, 64)	5×10^{-4}	1×10^{-5}	256	100	0.5	0.20	0.15	0.5	128	10	**94.2 ± 0.5**
2	(256, 128)	1×10^{-3}	1×10^{-5}	128	80	0.7	0.20	0.10	0.5	128	5	92.7 ± 0.6
3	(256, 128, 64)	1×10^{-4}	1×10^{-6}	256	100	0.5	0.10	0.10	0.7	256	5	94.0 ± 0.6
2	(128, 64)	5×10^{-4}	1×10^{-4}	512	50	0.5	0.10	0.05	0.4	64	5	91.8 ± 0.8
3	(256, 128, 64)	5×10^{-4}	1×10^{-5}	128	100	0.3	0.05	0.10	0.5	128	10	93.6 ± 0.7
3	(256, 128, 64)	5×10^{-4}	1×10^{-5}	256	100	0.7	0.20	0.20	0.5	128	5	88.3 ± 1.3

Table 3.6: Representative grid search results for CL's and Final Classifier's hyperparameters.

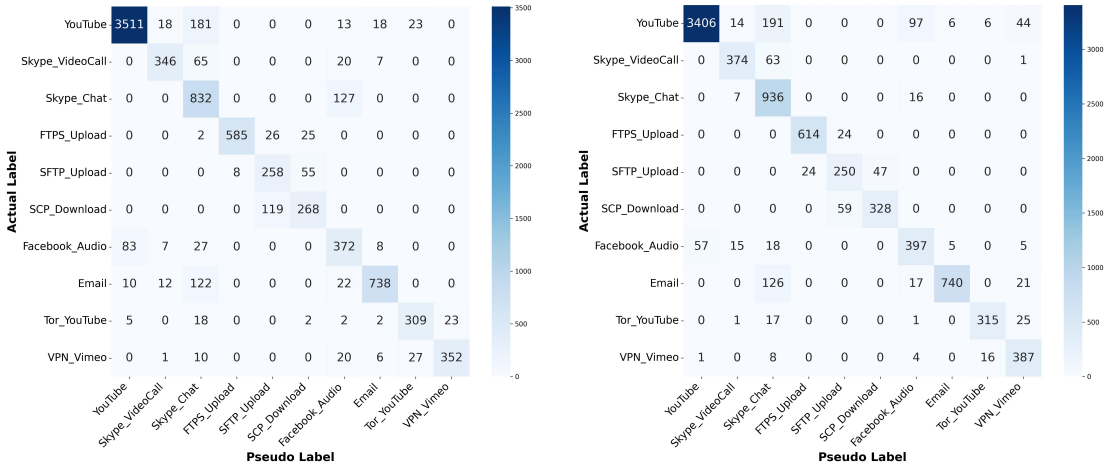
CL hyperparameters				Final classifier (MLP, weighted SCE)									Macro-F1 (%)
F-Folds	q	$w_{\min}$	γ	Depth	Hidden widths	LR	Weight decay	Batch	Dropout	Epochs	α	β	
5	0.70	0.20	4	3	(512, 256, 128)	1×10^{-4}	1×10^{-5}	128	0.30	50	0.3	2.0	**95.3 ± 0.5**
5	0.70	0.10	4	3	(512, 256, 128)	5×10^{-4}	1×10^{-5}	128	0.30	80	0.3	2.0	95.0 ± 0.7
3	0.70	0.20	2	2	(256, 128)	5×10^{-4}	1×10^{-5}	256	0.25	30	0.5	1.0	94.6 ± 0.7
5	0.60	0.30	8	3	(1024, 512, 256)	3×10^{-4}	1×10^{-4}	128	0.40	60	0.3	3.0	94.1 ± 0.8
10	0.70	0.20	4	3	(512, 256, 128)	1×10^{-3}	1×10^{-6}	64	0.20	60	0.1	2.0	93.9 ± 0.9
5	0.80	0.10	8	2	(256, 128)	1×10^{-3}	1×10^{-4}	256	0.20	30	0.1	1.0	93.2 ± 0.6

Finally, Table 3.7 presents the architecture and hyperparameters for the models utilized in the evaluation section.

Table 3.7: Hyperparameters of Models (final choices used in evaluation).

Hyperparameter	Autoencoder	TabCL	Confident Learning	Final Classifier
Model Architecture				
Hidden Layers	3	3	2	3
Neurons in 1st HL	256	256	128	512
Neurons in 2nd HL	128	128	64	256
Neurons in 3rd HL	64	64	–	128
Activation Function	ReLU	ReLU	ReLU	ReLU
Dropout Rate	0.10	0.10	0.20	0.30
Training Parameters				
Optimizer	ADAM	ADAM	ADAM	ADAM
Learning Rate	5×10^{-4}	5×10^{-4}	1×10^{-3}	1×10^{-4}
Batch Size	128	256	64	128
Number of Epochs	100	100	30	50
Weight Decay	1×10^{-5}	1×10^{-5}	–	1×10^{-5}
Specific Parameters				
Latent dimension	128	–	–	–
Constraint weight (ϕ)	0.5	–	–	–
Temperature τ_{cont}	–	0.50	–	–
Temperature τ_{cat}	–	0.20	–	–
Replacement rate (r)	–	0.15	–	–
Head mixing (λ)	–	0.50	–	–
Projection dim (per head)	–	128	–	–
Refresh interval (epochs)	–	10	–	–
F-folds (CL OOS probs)	–	–	5	–
Quantile (q)	–	–	0.70	–
Minimum weight ($w_{\min}$)	–	–	0.20	–
Logistic slope (γ)	–	–	4	–
α (CE term)	–	–	–	0.3
β (RCE term)	–	–	–	2.0

We applied AE and TabCL as SSL pretraining on the unlabeled dataset (8,684 flows), then fine-tuned with the small labeled set. AE achieved 87.17% pseudo-label accuracy, while TabCL reached 89.22%. A confidence–margin voting fusion of AE and TabCL produced 89.91%; the gain is limited by overlapping errors but slightly improves precision, aiding the CL stage. Fig. 3.5 shows confusion matrices between true labels and pseudo-labels, and Table 3.8 reports full metrics of AE, TabCL, and their fusion.



(a) Autoencoder.

(b) TabCL.

Figure 3.5: Confusion matrices of SSL models for pseudo-labelers.

Table 3.8: Performance metrics for SSL pseudo-labeling.

Model	**Precision**	**Recall**	**F1-score**	**Accuracy**
Autoencoder	88.30 ± 0.3	86.50 ± 0.7	87.20 ± 0.3	87.17 ± 0.3
TabCL	90.10 ± 0.5	88.60 ± 0.4	88.90 ± 0.2	89.22 ± 0.2
(AE $\oplus$ TabCL)	$\mathbf{90.40 \pm 0.4}$	$\mathbf{89.10 \pm 0.4}$	$\mathbf{89.30 \pm 0.3}$	$\mathbf{89.91 \pm 0.2}$

For training the final classifier, we use the fused pseudo-labels produced by the confidence–margin voting of AE and TabCL from the previous step. The fused pool attains an overall pseudo-label accuracy of 89.91%, implying that roughly 10% of labels are noisy. If this noise is not addressed and the final MLP is trained directly on the fused pseudo-labels, the test accuracy plateaus at 90.9%. To improve robustness, we apply traffic-adapted CL to estimate per-sample label reliability and down-weight likely errors before final training. With CL, the final MLP reaches an accuracy of $\mathbf{96.29}\%$. The confusion matrix for the final classifier is shown in Fig. 3.6, and per-class metrics are reported in Table 3.9, which shows uniformly high scores across most classes, with a few lower metrics because of feature similarity between these types of traffic.

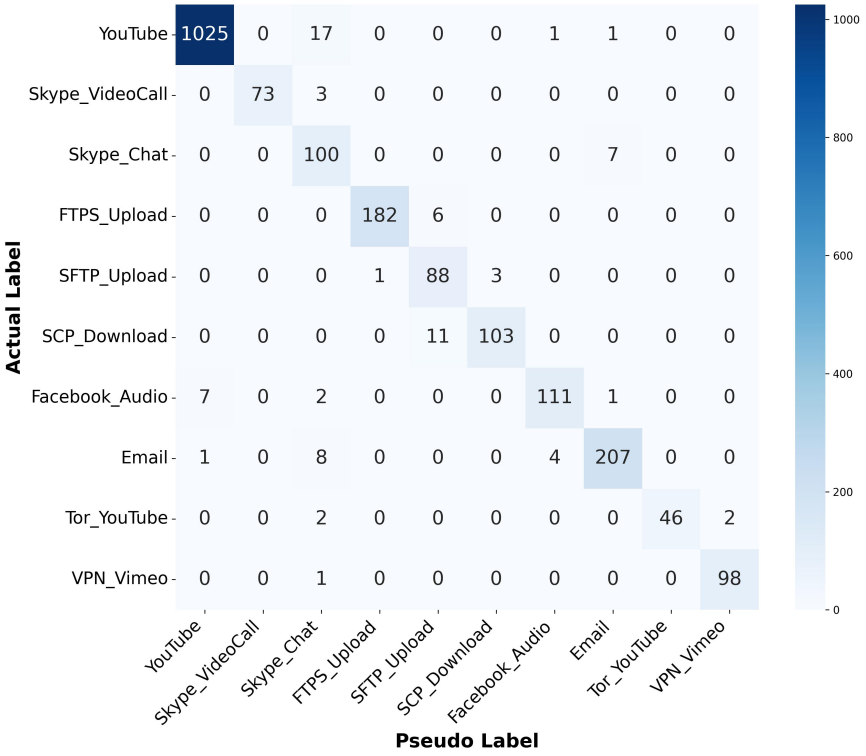


Figure 3.6: Confusion matrix for the MLP final classifier.

Table 3.9: Per-class performance of the final classifier.

Class	**Precision (%)**	**Recall (%)**	**F1 (%)**
YouTube	99.23	98.18	98.70
Skype_VideoCall	100.00	96.05	97.99
Skype_Chat	75.19	93.46	83.33
FTPS_Upload	99.45	96.81	98.11
SFTP_Upload	83.81	95.65	89.34
SCP_Download	97.17	90.35	93.64
Facebook_Audio	95.69	91.74	93.67
Email	95.83	94.09	94.95
Tor_YouTube	100.00	92.00	95.83
VPN_Vimeo	98.00	98.99	98.49
Overall (weighted avg)	**96.68**	**96.31**	**96.41**

Additionally, a bar chart in Fig. 3.7 illustrates the accuracy of the final classifier across individual classes, indicating that class-imbalance effects were effectively mitigated by the traffic-adapted CL stage.

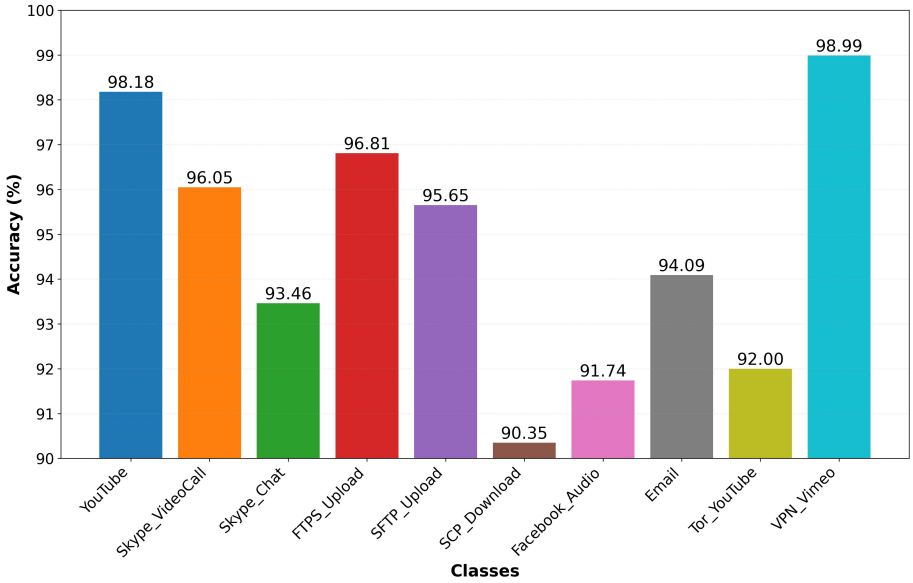


Figure 3.7: Per-class accuracy of final classifier.

In Table 3.10, we report a three-axis ablation. First, with the downstream CL+MLP stage fixed, the pseudo-label source is varied among AE, TabCL, and their fusion; fusion consistently outperforms either branch, indicating that higher-quality, lower-noise pseudo-labels directly improve downstream results. Second, for each pseudo-label source (AE, TabCL, fusion), CL is progressively upgraded from a mean-threshold filter to a traffic-adapted pipeline that adds quantile thresholds, logistic weighting, and BRC; every step improves performance, confirming the advantage of the traffic-aware CL. Finally, separating components confirms that neither SSL alone (without CL) nor CL alone (without SSL pretraining, just pseudo-labeling with a small labeled dataset) achieves the best outcomes; combining SSL with CL is required to obtain peak accuracy and macro-F1 with a small labeled set.

Table 3.10: Ablation of the final stage.

Setting	Macro-F1 (%)	Accuracy (%)
Pseudo-label source (for CL+MLP stage)		
AE pseudo-labels + CL	94.3 ± 0.4	94.2 ± 0.4
TabCL pseudo-labels + CL	95.7 ± 0.3	95.6 ± 0.3
Fusion (AE $\oplus$ TabCL) + CL	$\mathbf{96.43 \pm 0.3}$	$\mathbf{96.29 \pm 0.3}$
CL variants (on AE pseudo-labels)		
No-CL (direct training)	88.4 ± 0.6	88.4 ± 0.6
Mean threshold (original CL)	93.0 ± 0.4	92.9 ± 0.4
Quantile threshold only	93.3 ± 0.3	93.2 ± 0.3
+ Logistic weighting	93.8 ± 0.3	93.8 ± 0.2
+ BRC (full CL)	$\mathbf{94.3 \pm 0.4}$	$\mathbf{94.2 \pm 0.4}$
CL variants (on TabCL pseudo-labels)		
No-CL (direct training)	90.1 ± 0.5	90.3 ± 0.5
Mean threshold (original CL)	94.8 ± 0.3	94.8 ± 0.3
Quantile threshold only	95.2 ± 0.3	95.3 ± 0.3
+ Logistic weighting	95.5 ± 0.3	95.5 ± 0.3
+ BRC (full CL)	$\mathbf{95.7 \pm 0.3}$	$\mathbf{95.6 \pm 0.3}$
CL variants (on Fusion pseudo-labels)		
No-CL (direct training)	91.0 ± 0.5	90.9 ± 0.5
Mean threshold (original CL)	95.0 ± 0.4	95.2 ± 0.4
Quantile threshold only	95.4 ± 0.3	95.4 ± 0.3
+ Logistic weighting	96.0 ± 0.2	95.9 ± 0.2
+ BRC (full CL)	$\mathbf{96.43 \pm 0.3}$	$\mathbf{96.29 \pm 0.3}$
Component contribution		
Only SSL (Fusion, no CL)	91.0 ± 0.5	90.9 ± 0.5
Only CL (no SSL pretraining)	85.7 ± 0.6	85.8 ± 0.7
SSL + CL (ours)	$\mathbf{96.43 \pm 0.3}$	$\mathbf{96.29 \pm 0.3}$

Here, computed measurements are reported for both SSL pretraining and the CL with the final training, averaged across 10 seeds. According to Table 3.11, AE is efficient at 31.19 s/round; 1.2 GB peak VRAM, whereas TabCL is heavier at 107.86 s/round ($\approx$3.5$\times$ AE) and 1.5 GB due to contrastive multi-view and similarity computations. The CL plus final MLP stage is minimal (24.85 s/round, 0.90 GB; 0.018 M parameters) and achieves 31,240 flows/s. Pseudo-labeling itself is fast (AE: 93,464 flows/s; TabCL: 88,409 flows/s), indicating SSL pretraining, not CL/final training, dominates runtime. All stages fit comfortably on a 15 GB T4 and keep peak memory below 2 GB, indicating they are deployable on commodity GPUs. In terms of scalability, throughput scales near-linearly with the number of flows, supporting larger datasets. This low overhead enables deployment

in real-world scenarios, such as edge networks for online traffic analysis. SSL pretraining can be parallelized across workers or scheduled offline, while the CL & final MLP update remains quick and memory-frugal for frequent refreshes. Overall, the compute profile demonstrates efficiency (minutes per round) and resource frugality for scalable NTC applications.

Table 3.11: Compute profile of SSL and CL/final training.

Metric	**AE**	**TabCL**	**CL & Final MLP**
Time per round (s)	31.19	107.86	24.85
Training throughput (flows/s)	27,861	8,092	31,240
Pseudo-labeling (flows/s)	93,464	88,409	–
Peak GPU memory (GB)	1.20	1.50	0.90
Average GPU utilization (%)	32.2	43.1	28.7
Trainable parameters (M)	0.032	0.069	0.018

Figure 3.8 shows how CL adapts per-class decision thresholds to pseudo-label reliability. The dashed gray line is the class mean (original CL), while the magenta solid line is the quantile threshold $t_j^{0.7}$. For clean classes (YouTube, FTPS_Upload, Email), histograms peak near $s_i \approx 0.9$–1.0 with small MAD (≈ 0.03–0.05), and correspondingly high thresholds $t_j^{0.7} \in [0.90, 0.92]$. In contrast, classes that were noisier in the SSL (e.g., Skype_Chat, SFTP_Upload, and Facebook_Audio) show broader, left-shifted distributions with larger MAD (≈ 0.07–0.11) and noticeably lower thresholds, with $t_j^{0.7}$ moving to 0.74–0.82. This leftward adjustment avoids over-pruning hard classes by accepting lower confidence when assigning weight. In clean classes, right-skew places the mean left of the quantile, so using $t_j^{0.7}$ is less sensitive to low-confidence tails; in noisy classes, the quantile naturally tracks lower. These adaptive thresholds feed our CL weighting. As a result, the final classifier trained with CL improves from 90.9% (no CL) to 96.29% accuracy. The largest per-class gains occur exactly where Fig. 3.8 indicates higher noise and lower $t_j^{0.7}$ (e.g., Skype_Chat, SFTP_Upload, and Facebook_Audio), confirming that the performance gap between SSL pseudo-labeling and the final model is attributable to the CL stage.

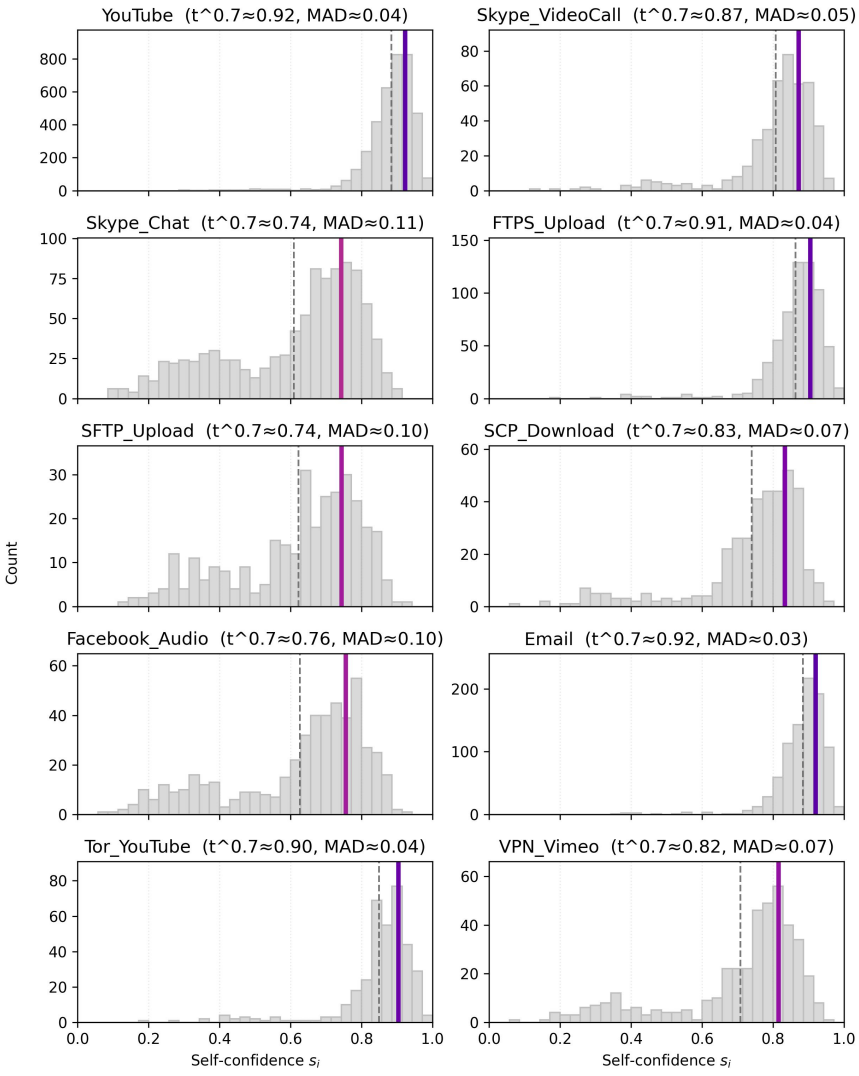


Figure 3.8: Per-class distributions of the self-confidence for pseudo-labeled flows.

To validate performance, we compare the framework with five baselines covering supervised and SSL settings: MTEFU [55], TFE-GNN [52], FlowPic [67], ET-BERT [65], and an SSL method for encrypted traffic [26]. For fairness, supervised baselines were trained on the self-generated labeled data and evaluated on the same splits, while SSL baselines used the self-generated corpus as unlabeled, plus the same small labeled set. As shown in Table 3.12, our SSL+CL pipeline attains 96.29% accuracy, surpassing FlowPic (93.73%) and ET-BERT (91.64%) and outperforming

TFE-GNN and multitask fusion, while using far fewer labels. Beyond the headline accuracy, our framework operates end-to-end—from raw packet capture to flow aggregation, pseudo-labeling, noise filtering, and final classification—so it can be deployed in realistic "low-label" settings rather than only curated benchmarks. Evaluations on our controlled, end-to-end corpus and on the public ISCX subset indicate that the method scales to thousands of flows and transfers with minimal tuning, suggesting good generalization while directly addressing label scarcity through SSL and CL.

Table 3.12: Performance metrics of our model and state-of-the-art methods

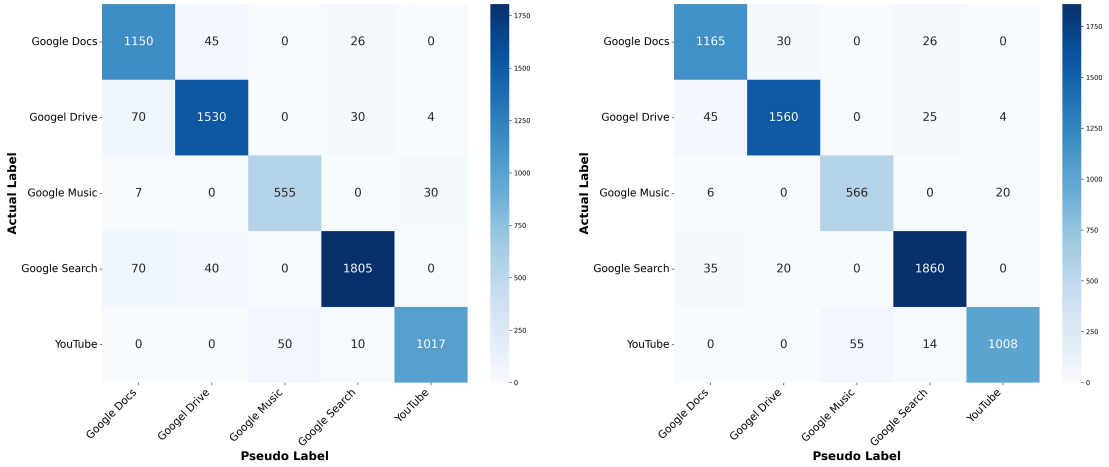
Methods	**Authors & Years**	**Precision**	**Recall**	**F1-score**	**Accuracy**
Multitask learning fusion	L. Liu et al. 2024 [55]	90.24	91.2	90.05	92.76
TFE-GNN	H. Zhang et al. 2023 [52]	94.21	93.91	93.88	93.45
SSL FlowPic	E. Horowicz et al. 2024 [67]	92.80	95.22	93.5	93.73
ET-BERT (flow-level)	X. Lin et al. 2022 [65]	93.40	94.01	93.49	91.64
NTC with SSL	M. Towhid et al. 2023 [26]	88.4	89.02	87.56	90.56
Our Method	Ours	**96.68**	**96.31**	**96.41**	**96.29**

3.5.4 Results on UCDavis–QUIC Dataset

To stress-test generalization beyond our self-generated/ISCX setting, we further evaluate on the UCDavis–QUIC dataset. In keeping with our "low-label" protocol, we treat 5% of flows per class as the small labeled set and the remaining 95% as unlabeled. Hyperparameters are selected exactly as in the previous subsection (stratified 5-fold CV on the small labeled split; three seeds per candidate; select by Macro-F1). Given the lower number of classes (5) and a more homogeneous feature schema, only minor changes were adopted: based on grid search, for the AE we use a slightly smaller latent size (64) to curb overfitting; in TabCL we reduce augmentation intensity (replacement rate $r = 0.10$) and temperature ($\tau_{\text{cont}} = 0.4$, $\tau_{\text{cat}} = 0.15$) to reflect the smoother intra-class structure; in CL we raise the quantile threshold to $q = 0.75$ and use a steeper logistic slope $\gamma = 6$ to be more selective with residual noise.

We apply both SSL branches independently on the unlabeled QUIC flows and then fine-tune on the 5% labeled split. Pseudo-label performance on the full dataset is strong: the AE reaches 94.07% accuracy, TabCL improves to 95.65%, and their fusion gives a further lift to 96.11%. Class-wise

confusion patterns for AE and TabCL are shown in Fig. 3.9. Aggregate metrics of SSL pseudo-labeling are detailed in Table 3.13.



(a) Autoencoder.

(b) TabCL.

Figure 3.9: UCDavis–QUIC: confusion matrices of SSL pseudo-labelers.

Table 3.13: UCDavis–QUIC: SSL pseudo-labeling metrics.

Model	**Precision (%)**	**Recall (%)**	**F1 (%)**	**Accuracy (%)**
Autoencoder	94.16 ± 0.28	94.07 ± 0.26	94.09 ± 0.24	94.07 ± 0.22
TabCL	95.70 ± 0.22	95.65 ± 0.21	95.66 ± 0.21	95.65 ± 0.20
(AE $\oplus$ TabCL)	$\mathbf{96.20 \pm 0.20}$	$\mathbf{96.10 \pm 0.20}$	$\mathbf{96.15 \pm 0.19}$	$\mathbf{96.11 \pm 0.18}$

Using the same workstation as in prior measurements, SSL round times were collected for the smaller QUIC corpus (6,439 flows, previously 8,684). As expected, both branches are faster: AE averages 22.21 s/round, while TabCL averages 51.95 s/round. Peak GPU memory is modest for both (1.0 GB for AE and 1.3 GB for TabCL), which aligns with the near-linear throughput scaling observed in the preceding subsection.

Next, the combined pseudo-labeled dataset is fed into the traffic-adapted CL module and used to train the final MLP classifier. This configuration attains $98.76\% \pm 0.10$ accuracy. The confusion matrix appears in Fig. 3.10; detailed per-class metrics are summarized in Table 3.14. Results are uniformly high, with the remaining errors largely arising from similar services such as Music and YouTube or Docs and Search.

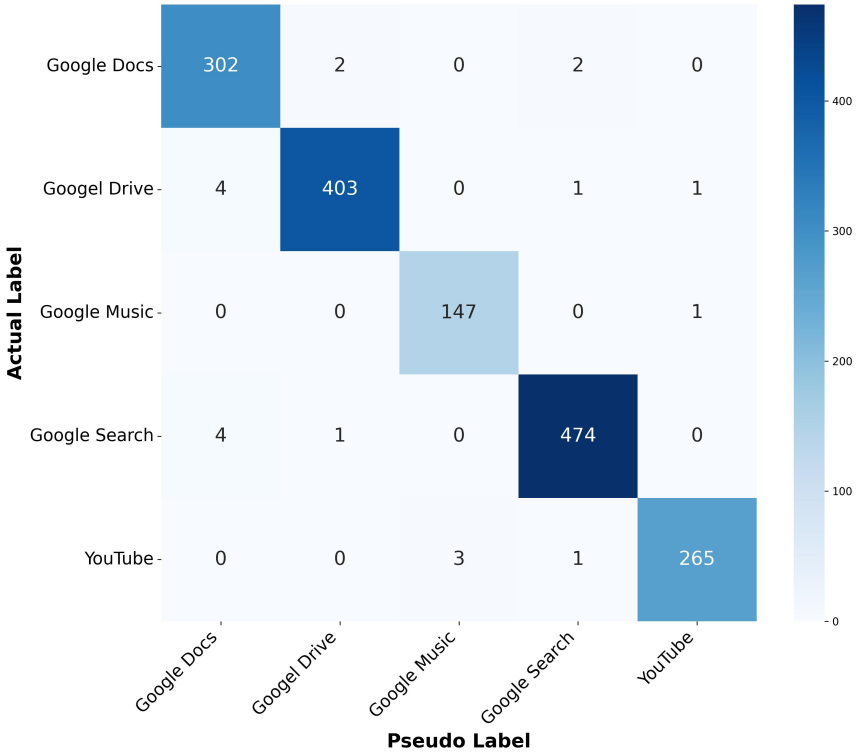


Figure 3.10: UCDavis–QUIC: confusion matrix of the final MLP with CL.

Table 3.14: UCDavis–QUIC: per-class performance of the final classifier.

Class	**Precision (%)**	**Recall (%)**	**F1 (%)**	**Accuracy (%)**
Google Docs	97.49	98.77	98.09	98.69
Google Drive	99.20	98.53	98.90	98.59
Google Music	97.67	98.99	98.32	98.99
Google Search	99.22	98.96	99.14	99.06
YouTube	99.25	98.51	98.79	98.33
Overall (weighted)	**98.85**	**98.74**	**98.69**	**98.76**

Table 3.15 summarizes three ablations under the same protocol as before (means over 10 seeds). (i) Pseudo-label source. Holding the downstream CL+MLP fixed, (ii) CL variants based on AE, TabCL, and Fusion pseudo-labels. Starting from "No-CL", progressively adding quantile thresholds, logistic weighting, and our BRC step yields monotonic gains, evidence that the traffic-adapted CL is more effective than the original mean-threshold heuristic. (iii) Component contribution that shows SSL without CL and CL without SSL pretraining are both inadequate; combining SSL with CL is necessary to reach the highest accuracy and macro-F1 with a 5% label budget.

Table 3.15: Ablation on UCDavis–QUIC.

Setting	Macro-F1 (%)	Accuracy (%)
Pseudo-label source (for CL+MLP)		
AE pseudo-labels + CL	97.33 ± 0.15	97.30 ± 0.15
TabCL pseudo-labels + CL	98.11 ± 0.12	98.05 ± 0.12
Fusion (AE $\oplus$ TabCL) + CL	$\mathbf{98.85 \pm 0.10}$	$\mathbf{98.76 \pm 0.10}$
CL variants (on AE pseudo-labels)		
No-CL (direct training)	94.15 ± 0.22	94.12 ± 0.22
Mean threshold (original CL)	96.80 ± 0.16	96.76 ± 0.16
Quantile threshold only	96.95 ± 0.16	96.90 ± 0.15
+ Logistic weighting	97.17 ± 0.12	97.13 ± 0.13
+ BRC (full CL)	$\mathbf{97.33 \pm 0.15}$	$\mathbf{97.32 \pm 0.15}$
CL variants (on TabCL pseudo-labels)		
No-CL (direct training)	95.75 ± 0.20	95.70 ± 0.20
Mean threshold (original CL)	97.69 ± 0.13	97.64 ± 0.13
Quantile threshold only	97.84 ± 0.14	97.82 ± 0.14
+ Logistic weighting	97.98 ± 0.12	97.89 ± 0.11
+ BRC (full CL)	$\mathbf{98.11 \pm 0.12}$	$\mathbf{98.05 \pm 0.12}$
CL variants (Fusion pseudo-labels)		
No-CL (direct training)	96.22 ± 0.18	96.19 ± 0.18
Mean threshold (original CL)	98.31 ± 0.12	98.25 ± 0.12
Quantile threshold only	98.47 ± 0.12	98.42 ± 0.12
+ Logistic weighting	98.66 ± 0.11	98.62 ± 0.11
+ BRC (full CL)	$\mathbf{98.85 \pm 0.10}$	$\mathbf{98.76 \pm 0.10}$
Component contribution		
Only SSL (Fusion, no CL)	96.24 ± 0.18	96.18 ± 0.18
Only CL (no SSL pretraining)	93.00 ± 0.22	92.90 ± 0.27
SSL + CL (ours)	$\mathbf{98.85 \pm 0.10}$	$\mathbf{98.76 \pm 0.10}$

Table 3.16 compares our approach with three representative baselines on UCDavis–QUIC. FlowPic's QUIC "script" setting reports 98.3% accuracy; the multitask fusion model reports $\approx 94.7\%$ for "Class" prediction; the SSL method of Towhid & Shahriar reports $\approx 98\%$ depending on the pretraining ratio. As seen, our SSL + CL pipeline attains the strongest overall scores on this dataset as well, reinforcing that the framework generalizes across three datasets (in two evaluation settings) and remains competitive even against specialized supervised or SSL baselines.

Table 3.16: UCDavis–QUIC: comparison our model with state-of-the-art methods.

Methods	Authors & Years	Precision	Recall	F1-score	Accuracy
Multitask learning fusion	L. Liu et al., 2024 [55]	94.80	94.58	94.69	94.67
SSL FlowPic (QUIC)	E. Horowicz et al., 2024 [67]	98.76	99.66	98.71	98.30
NTC with SSL	M. Towhid et al., 2023 [26]	98.12	97.93	98.03	98.01
Our Method	**Ours**	**98.85**	**98.74**	**98.69**	**98.76**

3.6 Conclusion

This chapter presented a label-efficient, centralized framework for NTC that pairs two SSL branches (constraint-consistent AE and TabCL with class-conditioned, constraint-preserving views and dual-head temperatures) with a traffic-adapted CL stage and a weighted SCE final classifier. The SSL branches generate high-quality pseudo-labels, optionally fused via a confidence–margin rule; CL then denoises pseudo-labels using per-class quantile thresholds, logistic weighting, and balanced retention. The approach demonstrates robust performance across datasets, attaining 96.29% accuracy on the self-generated and ISCX combination and 98.76% on UCDavis–QUIC.

While effective in centralized settings, the framework faces two practical limitations in data privacy and distributed environments, where raw data centralization is infeasible. The next chapter addresses these aspects by extending the method to a federated setting (privacy-enhancing, non-IID/imbalance aware) while preserving the accuracy benefits demonstrated here.

Chapter 4

FedSSL-NTC: A Robust Federated Self-Supervised Learning Framework for Network Traffic Classification Under Privacy Constraints

4.1 Introduction

This chapter represents the second major contribution of this thesis. Building on the foundational challenges from Chapter 1 and the literature gaps identified in Chapter 2, Chapter 3 presented a novel centralized framework that successfully addressed label scarcity and pseudo-label noise using a traffic-adapted SSL and CL pipeline. However, as noted in its conclusion, that approach does not address the critical, real-world challenges of data privacy, non-IID distributions, and class imbalance inherent to distributed networks. In real deployments, raw traffic flows are distributed across sites (edge devices, gateways, and domains) and cannot be centralized due to privacy, regulatory, or operational constraints. We therefore design and evaluate a federated variant that keeps data local while still leveraging a large unlabeled dataset and a small central labeled dataset. This chapter directly tackles these remaining gaps by extending our pipeline into a robust, privacy-aware

federated framework. We present FedSSL-NTC, a complete federated solution that achieves near-centralized accuracy while managing data heterogeneity and protecting client privacy. This chapter is based on material from our paper [94], which was published in the IEEE Open Journal of the Communications Society (OJ-COMS).

This chapter details the development of our federated framework. We first formalize the federated problem statement, its assumptions, and notation. We then present the complete system design of FedSSL-NTC, including its two-phase architecture for federated pre-training and confidence-aware classification. This section also details the specific mechanisms used to handle non-IID data (FedProx), class imbalance (class weights), quantity skew (SSW FedAvg), and privacy (SecAgg). Finally, we provide a comprehensive experimental evaluation on our two dataset combinations, demonstrating the framework's robustness and comparing its performance against the centralized model from Chapter 3 and other state-of-the-art federated methods.

4.2 Problem Statement, Assumptions, and Notation

In the federated setting for NTC, we assume a distributed environment with N clients (e.g., edge nodes or network sites). Each client $c \in \{1, \ldots, N\}$ holds a private, unlabeled local flow dataset D_l^c. A central server has access to a small, trusted labeled seed set D_s. The global objective is to train a K-class classifier $\mathcal{C}_{\text{final}}$ that learns from all distributed data without centralizing the raw, privacy-sensitive traffic flows.

This federated approach is motivated by the need to enhance privacy by keeping data local, which avoids centralizing sensitive information. However, while raw data remains local, exchanging model updates still poses potential privacy risks, as these updates could indirectly leak information about a client's local data. This motivates additional privacy-enhancing techniques. Beyond privacy, the federated setting introduces significant distributional challenges, as client data is often non-IID and suffers from class imbalance [34, 35]. These issues can destabilize training and degrade model accuracy, and must be explicitly managed.

Furthermore, this distributed framework must address the same challenge of label scarcity as

the centralized one. As established in Chapter 3, pseudo-labels generated via SSL inevitably contain noise, and training directly on them can degrade final classifier performance. Therefore, a successful federated framework must simultaneously manage privacy leakage, non-IID data, class imbalance, and be robust to this pseudo-label noise. Table 4.1 lists the key symbols and notations used throughout this chapter.

Table 4.1: List of Symbols

Symbol	Description
D_s	Small labeled dataset
D_l^c	Local unlabeled dataset of client c
$\mathbf{x}$	Input feature vector
y, $\tilde{y}$	True label, pseudo-label
K	Number of classes
N	Number of clients
$\mathcal{S}_r$	Set of participating clients
R	Communication rounds
$\mathcal{E}$, $\mathcal{C}$	Encoder backbone; classifier head
$\mathcal{L}$, η	Loss function, Learning rate
θ_g	Global model parameters in phase I
θ_c	Local model parameters of client c in phase I
θ'_c	Local model parameters in phase II
θ'_g	Global model parameters in phase II
n_c	Number of local samples used by client c
w_i	CL-derived per-sample weight on client c
$w_c[k]$	Class weight for class k on client c
μ	Proximal term coefficient in FedProx
∇	Gradient operator
α, β	Weight for forward/reverse cross-entropy in SCE
$\mathbf{u}_c$	Size-weighted update
ν_c	Sample-count scalar appended in SecAgg
$\mathbf{v}_c$	Augmented vector in SecAgg
$\widetilde{\mathbf{v}}_c$	Masked augmented vector sent by client c
$\mathbf{r}_{cj}$	Pairwise mask between clients c and j
T	Secret-sharing threshold for dropout handling

4.3 FedSSL-NTC System Design

FL is a decentralized machine learning paradigm that allows multiple clients to collaboratively train a model without sharing their raw data. In FL, each client trains a local model on its own

data and sends only the model updates (e.g., gradients or weights) to a central server. The server aggregates these updates to improve a global model, which is then redistributed to the clients for further training. This process is repeated over multiple rounds until the model converges. FL offers several advantages, particularly in scenarios where data privacy is a concern. By keeping data local to each client, FL avoids the need to centralize sensitive information, thus enhancing privacy. Additionally, FL can leverage diverse, distributed datasets, potentially leading to more robust and generalizable models. This is especially relevant for network traffic classification, where data may be collected from various networks with different characteristics.

This section presents the design of FedSSL-NTC, a two-phase federated pipeline that directly addresses the challenges outlined in the previous section. The pipeline is designed to: (i) build robust representations from abundant unlabeled flows via local SSL to generate pseudo-labels, and (ii) refine these pseudo-labels at the client-side using the traffic-adapted CL framework introduced in Chapter 3, before training a global classifier through a privacy-enhanced federated process. This process ensures robustness to non-IID data and class imbalance by incorporating FedProx [38], a class-weighted loss function, and SSW FedAvg. Furthermore, this framework is made privacy-aware by integrating the SSW FedAvg mechanism with a tailored SecAgg protocol [33].

The server coordinates the two phases. In Phase I, a shared SSL encoder is pretrained in a standard FL loop under FedProx and then fine-tuned on the small labeled seed set at the server; the resulting head is broadcast so that each client can pseudo-label its local unlabeled flows. In Phase II, clients apply CL locally to obtain per-sample weights for each flow. This traffic-adapted CL reduces the influence of uncertain pseudo-labels on-device, without sharing probabilities, thresholds, or weights with the server. Then, clients train the final classifier with a class-weighted Symmetric Cross Entropy (SCE) under a FedProx regularizer. To protect updates from an honest-but-curious server, we incorporate a SecAgg mechanism, which ensures the server only recovers the sum of client updates without accessing any individual client's contribution. Concretely, clients hide their updates with pairwise masks, and a threshold secret sharing scheme enables the server to remove masks contributed by dropped clients without learning any individual's update. Our approach tailors SecAgg to the SSW FedAvg method by augmenting client updates with data counts, enabling secure weighted aggregation. Figure 4.1 illustrates the architectural components of the FedSSL-NTC

framework.

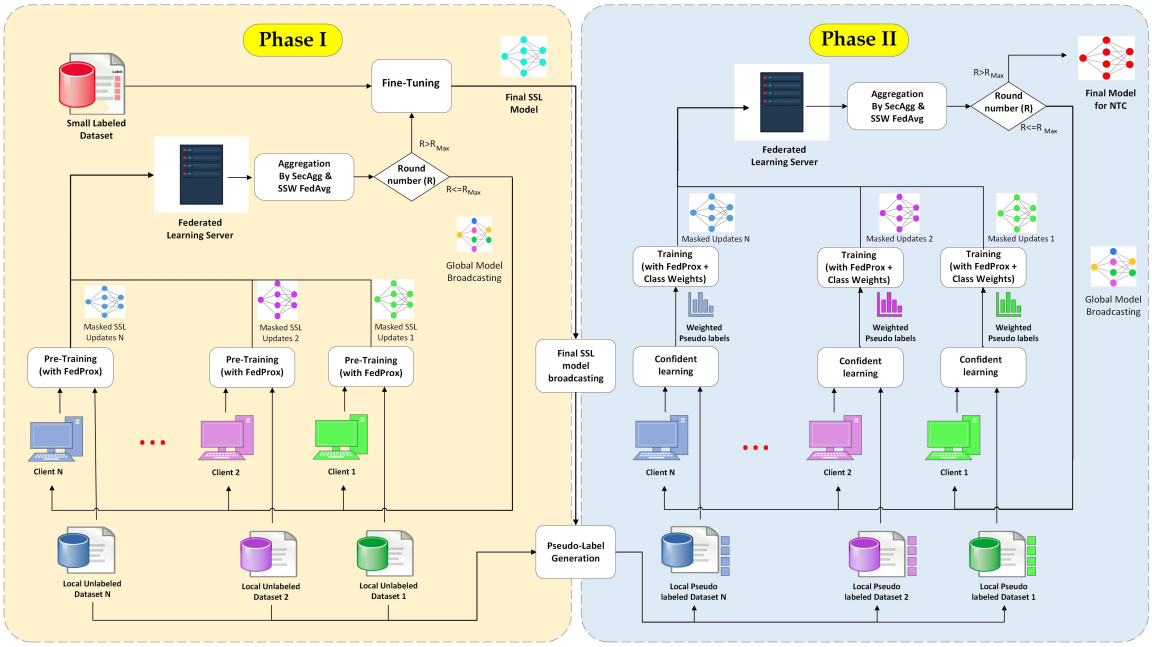


Figure 4.1: Architectural components of FedSSL-NTC.

4.3.1 Phase I: Federated SSL Pretraining and Pseudo-Labeling

Each round $r = 1, \ldots, R$, the server sends θ_g to a set of participating clients $\mathcal{S}_r$. Client c performs local SSL updates on D_l^c and returns its contribution to the SecAgg. Because network traffic is typically non-IID, we regularize local SSL with a proximal term to stabilize updates under heterogeneity:

$$\mathcal{L}_{\text{prox}}(\theta_c; \theta_g) = \frac{\mu}{2} \|\theta_c - \theta_g\|_2^2. \tag{41}$$

This proximal term provides a theoretical guarantee of convergence and discourages client drift in heterogeneous (non-IID) settings where standard FedAvg can diverge [38]. As μ increases, local updates are pulled toward θ_g, shrinking the deviation $\|\theta_c - \theta_g\|$ that appears in FedProx convergence analyses and improving stability when client objectives differ. In practice, a moderate μ balances local fitting (to each client's traffic mix) with global consistency. This ensures that local models do not overfit to their unique data distributions, providing a more stable and accurate global model. Let

$\mathcal{L}_{\text{SSL}}$ denote the branch-specific SSL loss (AE or TabCL). The local objective on client c is

$$\mathcal{L}_{\text{local}}^{\text{SSL}} \;=\; \mathcal{L}_{\text{SSL}} \;+\; \mathcal{L}_{\text{prox}}(\theta_c; \theta_g). \tag{42}$$

The $\mathcal{L}_{\text{SSL}}$ can be a loss function of AE or TabCL for pretraining, using the same traffic-adapted models from our centralized framework (detailed in Chapter 3). The AE reconstructs continuous fields (MSE) and categorical fields (cross-entropy) with a light algebraic-consistency penalty, making it suitable for NTC's heterogeneous tabular flows to produce plausible reconstructions. TabCL [36] uses class-conditioned replacements, a constraint-preserving projection for continuous features, and a dual-head NT-Xent objective tuned for continuous vs. categorical slices, enabling discriminative learning of traffic invariance. As justified in Chapter 3, AE and TabCL are selected for their balance: AE's generative strength handles NTC's feature reconstruction needs, while TabCL's contrastive approach suits class discrimination in federated, non-IID settings. Local optimization in the pre-training step follows standard gradient-descent updates on the corresponding loss:

$$\theta_c \leftarrow \theta_c - \eta \nabla_{\theta_c} \mathcal{L}_{\text{local}}^{\text{SSL}}. \tag{43}$$

Then, the server aggregates θ_c and refreshes θ_g if it does not use the SecAgg. For aggregation, the server applies SSW FedAvg:

$$\theta_g \;\leftarrow\; \sum_{c \in \mathcal{S}_r} \frac{n_c}{\sum_{j \in \mathcal{S}_r} n_j}\, \theta_c. \tag{44}$$

The SSW FedAvg mechanism offers a significant theoretical advantage over simple averaging (e.g., FedAvg). In NTC, data distribution is inherently heterogeneous in both class and traffic volume; for example, a core router (one client) may observe millions of flows, while an edge gateway (another client) observes only thousands. With simple FedAvg, few-sample clients (high-variance updates) can disproportionately perturb the global model. SSW FedAvg's theoretical advantage is that it reduces variance and stabilizes convergence under heterogeneous client sizes by weighting each client's contribution by its data volume n_c. This variance-reduction technique ensures that clients with more data (and thus more "valuable" and stable updates) have a proportionally larger influence, leading to faster, more stable convergence and a more accurate final model.

To enhance privacy during aggregation, clients apply SecAgg before sending updates to the server. Instead of sending θ_c directly to the server, clients participate in our tailored SecAgg protocol to protect their contributions. Our goal is to securely compute the SSW FedAvg, which requires the server to learn the weighted sum $\sum_{c\in\mathcal{S}_r} n_c\theta_c$ and the sum of total samples $\sum_{j\in\mathcal{S}_r} n_j$ without revealing any individual client's update θ_c or sample count n_c. Inspired by the SecAgg protocol [33], to realize SSW FedAvg, each client forms an augmented vector:

$$\mathbf{v}_c \;=\; \big[\; \underbrace{\mathbf{u}_c}_{n_c\,\theta_c} \;;\; \underbrace{\nu_c}_{n_c} \;\big], \tag{45}$$

whose first block has the same dimension as θ_c and the last entry is the scalar n_c. Each client then generates a private random mask and establishes pairwise random masks $\{\mathbf{r}_{cj}\}$ with other clients $j \in \mathcal{S}_r$ (e.g., using Diffie-Hellman key exchange) and expanded by a Pseudo-Random Generator (PRG) to the augmented length. Next, Client c masks $\mathbf{v}_c$ with pairwise random masks that are constructed such that $\mathbf{r}_{cj} = -\mathbf{r}_{jc}$, ensuring sum to zero over all clients:

$$\widetilde{\mathbf{v}}_c \;=\; \mathbf{v}_c \;+\; \sum_{j>c} \mathbf{r}_{cj} \;-\; \sum_{j<c} \mathbf{r}_{jc}. \tag{46}$$

To protect against client dropouts in SecAgg, each client also splits its private mask into N shares using Shamir's Secret Sharing (T-out-of-N threshold). Client c sends one share to each of the other $N-1$ clients. If some clients drop out, the server can remove only the masks associated with the dropped clients, without learning any individual update. The server receives only the masked vectors $\{\widetilde{\mathbf{v}}_c\}$ and computes their sum; pairwise masks cancel, yielding:

$$\sum_{c\in\mathcal{S}_r} \widetilde{\mathbf{v}}_c \;=\; \left[\; \sum_{c\in\mathcal{S}_r} n_c\,\theta_c \;;\; \sum_{c\in\mathcal{S}_r} n_c \;\right]. \tag{47}$$

From (47), the server updates the global parameters by secure weighted averaging:

$$\theta_g \;\leftarrow\; \frac{\sum_{c\in\mathcal{S}_r} n_c\,\theta_c}{\sum_{j\in\mathcal{S}_r} n_j}. \tag{48}$$

Equation (48) is algebraically identical to SSW FedAvg, but the computation is performed over

masked messages, so the server learns only the two totals in (47). Thus, SecAgg changes observability—not optimization—delivering privacy-aware aggregation with no loss in accuracy.

After R rounds, the server attaches a classifier head for fine-tuning. After fine-tuning $\mathcal{E}$+$\mathcal{C}$ on D_s, the server broadcasts the tuned model. In the next step, each client obtains $\tilde{y}$ for its D_l^c using either the AE branch, the TabCL branch, or a fusion of both. In line with centralized results in chapter 3, the fusion variant typically yields higher pseudo-label fidelity and, in turn, better downstream performance. The above steps are summarized in Algorithm 5.

Algorithm 5 Federated SSL for Pseudo-Labeling (with Secure Weighted Aggregation)

Input: Unlabeled datasets $\{D_l^c\}_{c=1}^N$, labeled dataset D_s, number of clients N, number of rounds R
Output: Pseudo-labeled dataset for each client
Initialize global SSL model with parameters θ_g
for each round $r \in \{1, \ldots, R\}$ **do**
    for each client $c \in \mathcal{S}_r = \{1, \ldots, N\}$ in parallel **do**
        Client c receives global model θ_g
        Client c trains local SSL model on D_l^c using (42) to get θ_c
        Client c constructs $\mathbf{v}_c = [\, n_c\theta_c;\ n_c \,]$ and mask it as in (46)
        Client c send masked $\widetilde{\mathbf{v}}_c$ to the server
    end for
    Server securely sums $\{\widetilde{\mathbf{v}}_c\}$; recover totals via mask cancellation / threshold shares
    Server updates θ_g by (48)
end for
Fine-tune $\mathcal{E} + \mathcal{C}$ on D_s and broadcast
for each client $c \in \mathcal{S}_r = \{1, \ldots, N\}$ in parallel **do**
    Generate pseudo-labels $\tilde{y}^c$ for D_l^c
end for

4.3.2 Phase II: Confidence-Aware Federated Classification

In phase II, clients first apply traffic-adapted CL locally and independently on $(D_l^c, \tilde{y}^c)$ to obtain per-sample weights w_i. This step is performed entirely on the client side, where each client uses its local pseudo-labeled data to estimate label reliability without sharing any information with the server or other clients. In the federated setting, this local execution of CL ensures that noise estimation and weight assignment remain private, aligning with FL's data locality principle. Similar to the centralized framework described in Chapter 3, to handle noisy pseudo-labels, the traffic-adapted CL process begins by computing out-of-sample predicted probabilities for each local sample through

cross-validation (CV) on the client's own data. From these, self-confidence scores are derived, representing the model's predicted probability for the assigned pseudo-label. Per-class thresholds are then set using quantiles of these scores to adapt to skewed distributions common in traffic data, providing robustness over mean-based approaches (original method of CL). These thresholds feed into a logistic weighting scheme that smoothly attenuates low-confidence samples while retaining all data. Next, a balanced retention step adjusts weights based on the confident joint matrix, which estimates clean label fractions and scales class masses accordingly to counter imbalance. By obtaining w_i, it mitigates the impact of noisy pseud-labels during training of the final classifier in phase II.

To address local class imbalances, each client calculates class weights, promoting fairness in underrepresented categories:

$$w_c[k] \;=\; \frac{1}{\mathrm{freq}_c(k) + \epsilon}, \qquad k = 1, \ldots, K, \tag{49}$$

where $\mathrm{freq}_c(k)$ is the count of pseudo-labeled samples of class k retained for training on client c and $\epsilon > 0$ avoids division by zero. In NTC, traffic is notoriously imbalanced (e.g., HTTPS flows often outnumber VoIP flows by orders of magnitude). Without this re-weighting, the local model would quickly overfit to the dominant classes, resulting in a global model (after aggregation) that performs poorly on minority, but often critical, traffic types. With inverse-frequency scaling, the effective per-class contribution is roughly $\mathrm{freq}_c(k) \cdot w_c[k] \approx 1$, which balances minority classes and forces the model to learn their features. This class weight is independent of SSW FedAvg: SSW mitigates skew across clients (by sample size), whereas $w_c[k]$ mitigates skew within a client (by class mix). Together, they control two independent sources of imbalance common in network traffic. Afterward, using predicted probabilities $\mathbf{p}_i = \mathcal{C}(\mathbf{x}_i)$, each client trains a local classifier with a robust loss (class-weighted SCE) combined with a FedProx term using the Phase I pseudo-labels:

$$\begin{aligned}\mathcal{L}_{\text{local}}^{\text{cls}} = \; w_i \Big[\alpha \, \mathrm{CE}_w(\mathbf{p}_i, \tilde{y}_i; w_c) \;+\; \beta \, \mathrm{RCE}(\tilde{y}_i, \mathbf{p}_i) \Big] \\ +\; \frac{\mu}{2} \, \|\theta'_c - \theta'_g\|_2^2.\end{aligned} \tag{50}$$

Here CE_w is the class-weighted cross-entropy; RCE is the reverse cross-entropy; $\alpha, \beta > 0$ are fixed coefficients. Class weights w_c in (49) mitigate skew within each client, while the proximal term limits drift toward client-specific optima under non-IID data. During Phase II, each client updates its classifier parameters using vanilla gradient descent on the local classification objective:

$$\theta'_c \leftarrow \theta'_c - \eta \nabla_{\theta'_c} \mathcal{L}^{\text{class}}_{\text{local}}. \tag{51}$$

In the next step, the server aggregates classifier parameters in the same privacy-enhancing secure protocol as in Phase I and independently. Each client forms $\mathbf{v}'_c = [\, n_c \theta'_c;\, n_c \,]$ (using their local sample count n_c for this phase), and masks it as in (46) (with new pairwise masks). Then, the server computes

$$\theta'_g \;\leftarrow\; \frac{\sum_{c \in \mathcal{S}_r} n_c\, \theta'_c}{\sum_{j \in \mathcal{S}_r} n_j}. \tag{52}$$

Algorithm 6 details the phase II procedure.

Algorithm 6 Confidence-Aware Federated Classification

Input: Pseudo-labeled datasets $(\{D_l^c\}_{c=1}^N, \tilde{y}^c)$, number of clients N, number of rounds R
Output: Trained global classifier $\mathcal{C}_{\text{final}}$
Initialize global classifier $\mathcal{C}_{\text{final}}$ with parameters θ'_g
for each client $c \in \{1, \ldots, N\}$ in parallel **do**
    Client c runs CL on $(D_l^c, \tilde{y}^c)$ to obtain w_i
    Client c computes class weights w_c using (49)
end for
for each round $r \in \{1, \ldots, R\}$ **do**
    for each client $c \in \{1, \ldots, N\}$ in parallel **do**
        Client c receives global model θ'_g
        Client c trains local classifier using (50)
        Client c constructs $\mathbf{v}'_c = [\, n_c \theta'_c;\, n_c \,]$, and mask it as in (46)
        Client c send $\widetilde{\mathbf{v}}'_c$ to the server
    end for
    Server securely sums $\{\widetilde{\mathbf{v}}'_c\}$ and updates θ'_g by (52)
end for

In FedSSL-NTC, hyperparameter selection is performed exclusively on D_s available on the server, as this is the only labeled data accessible in the framework, reflecting label-scarce scenarios. Hyperparameters fall into two categories: (i) model and learning parameters (e.g., learning rate, batch size, hidden layer, epochs, temperatures for TabCL, quantile for CL, and α and β for SCE

in the final classifier); and (ii) FL-specific parameters (e.g., proximal coefficient μ in FedProx and communication rounds R). To tune these, we first split D_s into a training and validation subset (e.g., 80% and 20%). For model-specific hyperparameters in Phase I (AE and TabCL), we apply stratified k-fold CV on the training subset, selecting those that maximize macro-F1 (averaged over folds and random seeds) via grid search on the server side, the same way as in chapter 3. Then these parameters will be shared with clients. With the model parameters fixed, we conduct a grid search over candidate values for μ (e.g., 0.001, 0.01, 0.1) while monitoring the fine-tuned model's accuracy on the validation subset. The R is determined by the validation accuracy converging. This same method is applied in Phase II: the k-fold CV determines the optimal CL and final classifier parameters, while we use grid search for μ and monitor convergence of final accuracy on the validation subset for R.

NTC data are naturally heterogeneous across sites (due to different mixes of applications, device populations, and time windows). Plain local training can overfit client-specific modes and push local optima far from the global solution. The FedProx penalty in (41) and (50) explicitly discourages large parameter deviations and is therefore effective at stabilizing convergence under non-IID data. At the same time, pseudo-labeled pools are typically imbalanced (e.g., dominant video or browsing classes). The per-class weights in (49) rebalance the loss so that minority classes remain competitive during local optimization. Together with CL weights $\{w_i\}$, these mechanisms reduce the impact of label noise and class skew on each client while enhancing privacy: not only do raw flows never leave the client site, but the integration of SecAgg also ensures that individual model parameters are shielded from the central server during aggregation, while having no impact on model accuracy, as the protocol cryptographically computes the exact aggregate sum. This decentralized architecture advances NTC by enabling privacy-aware, adaptive classification in distributed scenarios.

4.4 Experimental Evaluation and Results

This section outlines the experimental assessment of FedSSL-NTC, emphasizing its resilience in distributed settings. We evaluate the quality of federated self-supervised pseudo-labeling and the robustness of the federated classifier under statistical heterogeneity and class imbalance. We

measure performance through standard metrics such as precision, recall, F1-score, and accuracy, while highlighting comparisons to the centralized baseline from our prior chapter 3, FedSSL-NTC, and state-of-the-art methods for NTC.

4.4.1 Experimental Setup and Datasets

Self-generation and ISCX VPN-nonVPN dataset: For continuity with the centralized experiments, we use the same two data sources as in Chapter 3: (1) a large self-generated NetFlow-derived dataset (8,684 flows) that we treat as unlabeled, and (2) a small labeled subset from the ISCX VPN-nonVPN [14] corpus (538 flows) used as a seed labeled set for server-side fine-tuning. ISCX is a standard, widely used benchmark covering heterogeneous VPN and non-VPN (encrypted/plain) traffic. The flows in these two datasets are categorized into distinct application types to encompass diverse network behaviors, including multimedia exchanges (e.g., YouTube, Email, Skype_Chat), bulk data transfers (e.g., SCP_Download, FTPS_Upload, SFTP_Upload), latency-sensitive interactions (e.g., Facebook_Audio, Skype_VideoCall) , and VPN-based communications (e.g., Tor_YouTube, VPN_Vimeo). In our evaluation, the server retains the small labeled dataset (the 538 ISCX flows) solely for controlled fine-tuning. This mirrors real-world cases where a small trusted labeled set exists centrally while most data remains local and unlabeled. The self-generated dataset is intentionally distributed among five clients. This distribution is designed with unequal data sizes and class distributions to reflect real-world challenges, such as non-IID data and class imbalance. The data distribution across these clients is illustrated in the heatmap in Fig. 4.2.

To simulate natural client heterogeneity common in real networks, we partitioned the data to reflect both quantity skew (where client data sizes vary, with Client 1 holding 2.7× more data than Client 5) and label skew, where all clients share a dominant class (YouTube) but in varying proportions. Quantity skew is quantified by the Gini coefficient, calculated on client totals of 2604, 2083, 1389, 1649, and 958 flows for Clients 1–5, yielding Gini ≈ 0.18 (0 = perfect equality; 1 = total inequality). Label skew is measured by the average Total Variation Distance (TVD) between each client's label distribution and the global one, with $\text{TVD}_{\text{avg}} \approx 0.14$. This controlled setup tests the framework's robustness to varying data volumes and label imbalances while ensuring reproducibility.

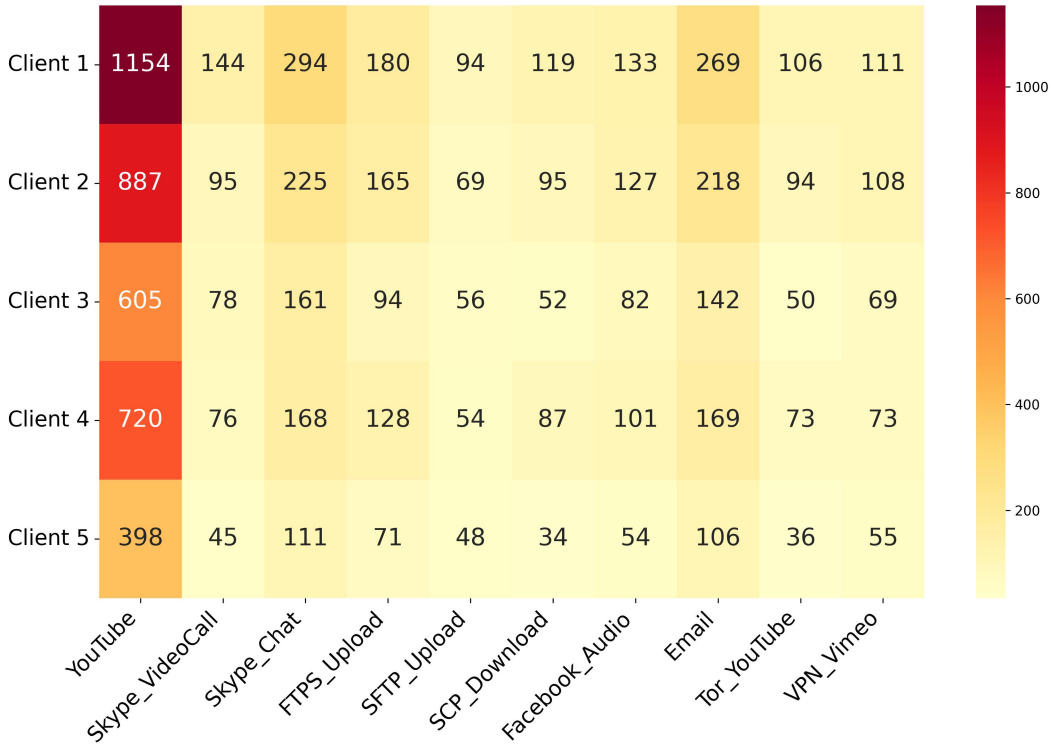


Figure 4.2: Class distribution heatmap across five clients (self-generated dataset).

UCDavis–QUIC Dataset: To broaden evaluation and assess generalization to modern encrypted traffic, we also use the UCDavis–QUIC dataset [39], which contains modern encrypted QUIC-only flows from five Google service classes (Google Music, Google Drive, Google Docs, YouTube, and Google Search). QUIC flows are derived from packet capture (PCAP) files and represented by 24 per-flow numerical statistics (plus an explicit flow-direction indicator) following the dataset's original schema. Consistent with our label-scarce setting, we retain a small stratified labeled subset of approximately 5% per class for server-side fine-tuning (323 flows in total) and treat the remaining 6,116 flows as unlabeled. The unlabeled portion is partitioned across four clients with unequal sizes and skewed class composition to emulate a non-IID federation. For this dataset, we simulated a more challenging pathological label skew to represent distinct user profiles. In contrast to the previous dataset, the quantity skew is low (Gini ≈ 0.130); however, the label distribution skew is high. For example, Client 1 is skewed toward "Google Search" (a 'researcher' profile), while Client 2 is heavily skewed toward "Google Drive" (a 'storage' profile). This significant difference

in the mix of labels is quantified by a high average TVD ≈ 0.184 between client and global distributions, confirming a highly non-IID partition. The resulting client–class distribution for QUIC is summarized visually in Fig. 4.3.

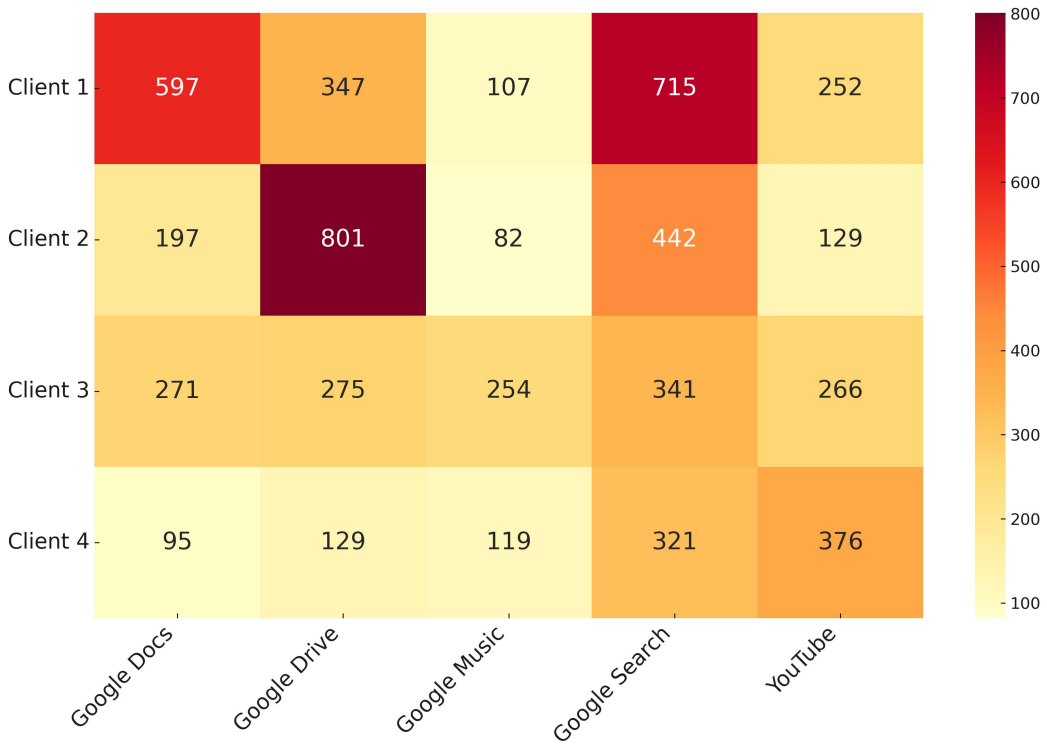


Figure 4.3: Class distribution heatmap across four clients (UCDavis–QUIC).

All experiments were conducted on a Linux workstation configured: Ubuntu 22.04.4 LTS with Python 3.12.11 and PyTorch 2.8.0+cu126, an Intel Xeon CPU @ 2.00 GHz, 12.7 GB system RAM, and a single NVIDIA Tesla T4 GPU (15.0 GB).

4.4.2 Results on Self-Generated and ISCX Datasets

We assess FedSSL-NTC in two stages: its federated SSL pseudo-labeling and its final, CL-denoised federated classifier. In training, the self-generated dataset was treated as unlabeled and split across clients, with ground-truth labels used only for evaluation to simulate label-scarce environments. To optimize hyperparameters in FedSSL-NTC, we conducted a 5-fold CV on the ISCX benchmark dataset. For each candidate setup, training was replicated three times using distinct random seeds, with accuracy reported as the mean ± 95% confidence interval. Optimal configurations

per component were chosen based on the highest accuracy. Given the extensive hyperparameters from the previous chapter 3.5, this analysis emphasizes FL-specific parameters. Table 4.2 illustrates the grid search outcomes for μ, α, and β as a sensitivity study, holding other parameters constant. Table 4.3 details the final model configurations applied in the evaluations.

Table 4.2: Sensitivity analysis of key hyperparameters for the Self-gen+ISCX dataset.

Hyperparameter	**Value Tested**	**Validation Accuracy (%)**
FedProx Term μ (Phase II)	0.001	94.14 ± 0.28
	0.01	**95.11 ± 0.15**
	0.1	94.74 ± 0.22
	1.0	93.92 ± 0.33
SCE Forward Weight (α)	0.1	94.88 ± 0.19
	0.3	**95.11 ± 0.17**
	0.5	94.46 ± 0.28
SCE Reverse Weight (β)	1.0	94.65 ± 0.30
	2.0	**95.11 ± 0.17**
	3.0	94.79 ± 0.2

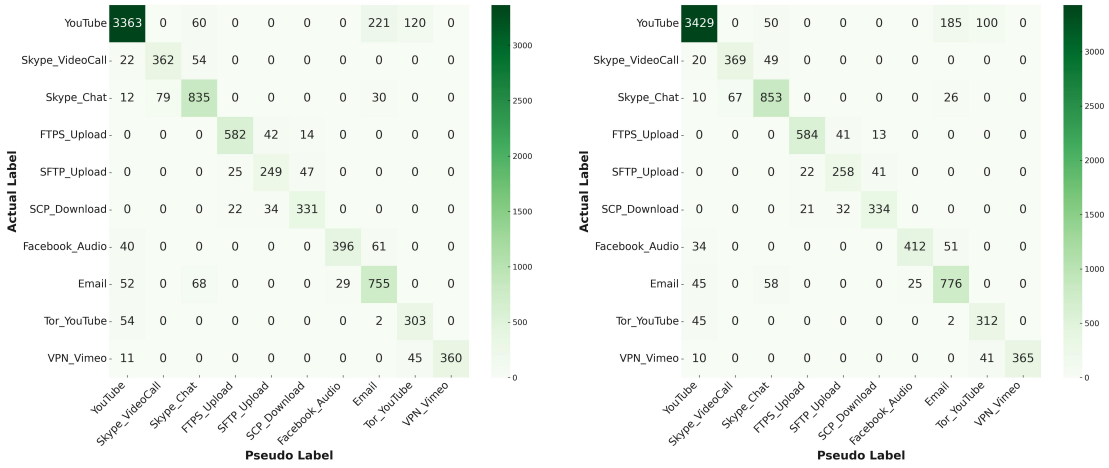
Table 4.3: Hyperparameters of Models in FedSSL-NTC.

	Phase I		**Phase II**	
Hyperparameter	**AE**	**TabCL**	**CL**	**Final Classifier**
Model Architecture				
Hidden Layers	3	3	2	3
Neurons in 1st HL	256	256	128	512
Neurons in 2nd HL	128	128	64	256
Neurons in 3rd HL	64	64	–	128
Dropout Rate	0.10	0.10	0.20	0.30
Activation Function	ReLU	ReLU	ReLU	ReLU
Training Parameters				
Learning Rate	5×10^{-4}	5×10^{-4}	1×10^{-3}	1×10^{-4}
Batch Size	128	256	64	128
Number of Epochs	100	100	30	50
Weight Decay	1×10^{-5}	1×10^{-5}	–	1×10^{-5}
Optimizer	ADAM	ADAM	ADAM	ADAM
FL Specific Parameters				
μ (Proximal term)	0.01	0.01	–	0.01
α (CE term)	–	–	–	0.3
β (RCE term)	–	–	–	2.0
R (Training Round)	20	20	–	20

In this part, we evaluate phase I, considering both the constraint-consistent AE and class-conditioned, constraint-preserving, dual-head TabCL methods. Each client performs 20 local pre-training rounds. In addition to reporting each branch separately, we also consider a lightweight fusion of AE and TabCL predictions, yielding a single pseudo-label per flow. Table 4.4 lists per-client and overall pseudo-labeling accuracy for the AE, TabCL, and their fusion. The observed overall accuracies are 86.82% (AE) and 88.62% (TabCL). The overall matrices in Fig. 4.4 are strongly diagonal, indicating consistent pseudo-label quality across classes; residual errors are concentrated among a few closely related classes.

Table 4.4: Accuracy of clients for adding Pseudo labels in the FL setting.

Clients	**Accuracy (%)**		
	AE	TabCL	Fusion (AE+TabCL)
Client 1	87.10	89.21	89.86
Client 2	87.02	89.11	89.50
Client 3	86.71	88.59	89.02
Client 4	85.93	87.83	88.67
Client 5	87.34	88.36	89.45
Overall	**86.82**	**88.62**	**89.30**



(a) Autoencoder (b) TabCL.

Figure 4.4: Pseudo-label confusion matrices in FL: (a) Autoencoder, (b) TabCL.

Given that the fusion method achieved higher pseudo-label accuracy in the previous phase, we used its pseudo-labeled dataset to train the final classifier in phase II. For evaluation by ground truth

labels, 25% of each client's (originally labeled) flows were reserved as a test set. We analyzed the impact of the total number of communication rounds on overall model accuracy, with the results detailed in Table 4.5. The peak accuracy of 95.88% is achieved at round 15. The model remains stable at 95.79% by round 20, confirming that 15-20 rounds are sufficient to reach full convergence.

Table 4.5: Final classifier's accuracy in 20 training rounds.

Rounds	**Accuracy (%)**
2	67.60
6	86.21
10	91.54
15	**95.88**
20	95.83

To quantify the contribution of each component within FedSSL-NTC, the comprehensive ablation study is detailed in Table 4.6. First, evaluating the pseudo-label source shows that the Fusion of AE and TabCL provides a measurable advantage. Second, an incremental build-up starting from a Simple FedAvg baseline demonstrates the cumulative benefit of adding SSW-FedAvg, FedProx, and finally Class Weights. Notably, the full model, including SecAgg also achieves 95.88% accuracy, proving it adds a robust privacy guarantee at zero cost to performance. Finally, a subtractive ablation confirms the individual importance of each piece. Removing CL and training on noisy pseudo-labels causes a significant accuracy drop, while removing either the Class Weights or FedProx also incurs substantial penalties, underscoring their vital role in handling class imbalance and non-IID data.

The confusion matrices for each client and the overall model, based on the test sets, are presented in Fig. 4.5, offering insights into both individual and collective performance. Figure 4.6 complements the ablation study by visualizing the convergence behavior of the final classifier. The full FedSSL-NTC model demonstrates the most stable and effective path to the lowest loss. In contrast, the model without FedProx clearly suffers from instability, exhibiting an erratic loss curve that converges to the highest level, which underscores its critical role in managing non-IID client drift. The model without Class Weights appears to converge fastest at the start; however, this reflects the model quickly learning only the majority classes and then getting stuck at a high loss, proving that

Table 4.6: Ablation study on FedSSL-NTC using the Self-gen+ISCX dataset.

Setting	Macro-F1 (%)	Accuracy (%)
1. Pseudo-label Source (with Full Model)		
AE-based Pseudo-Labels	93.91	93.72
TabCL-based Pseudo-Labels	94.96	94.75
Fusion (AE $\oplus$ TabCL)	**96.16**	**95.88**
2. Incremental Build-up (on Fusion Labels)		
Simple FedAvg (Baseline)	66.03	68.78
SSW-FedAvg	70.51	72.65
+ FedProx	91.46	91.12
+ Class Weights	**96.16**	**95.88**
+ SecAgg (Full Model)	**96.16**	**95.88**
3. Subtractive Ablation (on Fusion Labels)		
Full Model (Ours)	**96.16**	**95.88**
... without Class Weights	91.46	91.12
... without FedProx	88.84	88.38
... without SSW-FedAvg	92.49	92.10
... without CL (Noisy Labels)	88.97	89.63

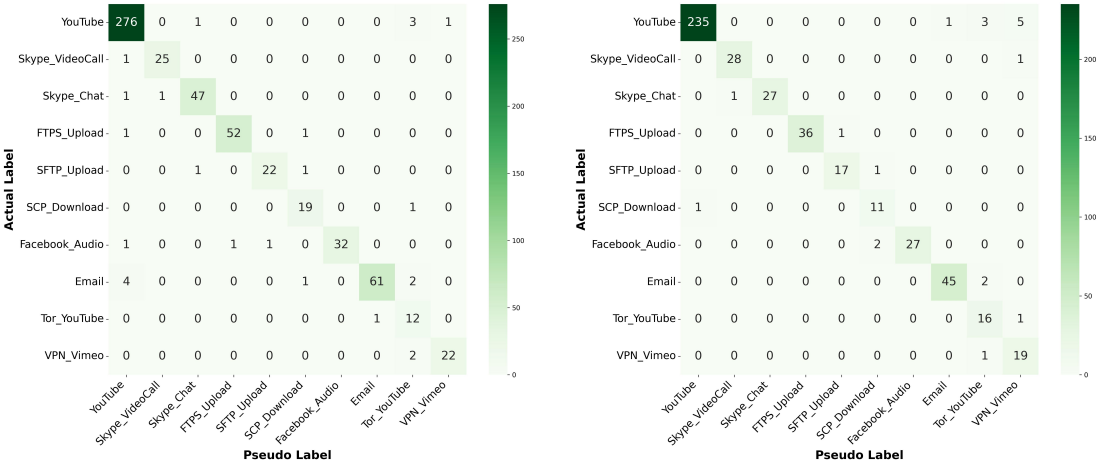
this component is essential for handling class imbalance.

The final accuracy and other performance metrics for each client and the overall model are summarized in Table 4.7.

Table 4.7: Performance metrics for final classifier in FedSSL-NTC.

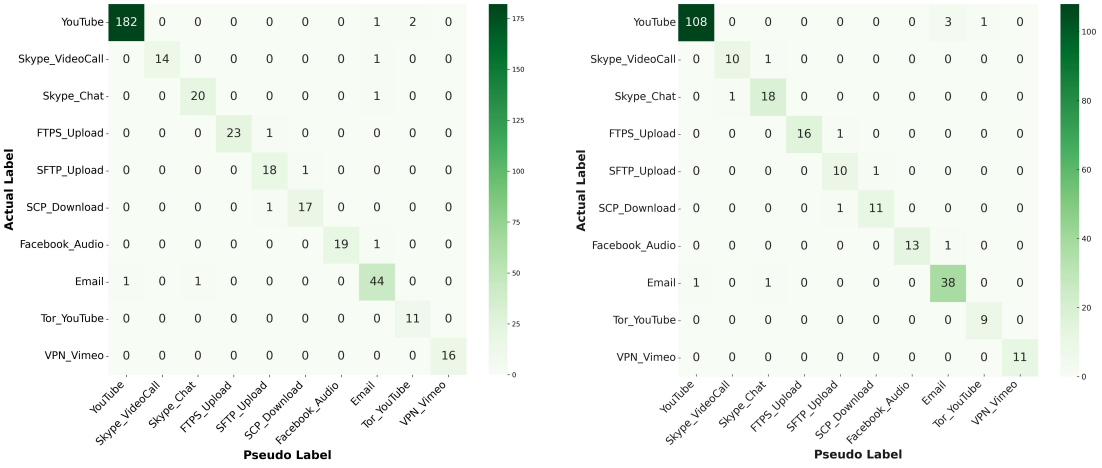
Client	Precision	Recall	F1-score	Accuracy
Client 1	96.12	95.19	95.65	95.62
Client 2	96.48	95.61	96.04	95.84
Client 3	97.03	96.18	96.60	96.55
Client 4	96.05	95.01	95.53	95.31
Client 5	96.80	95.76	96.27	96.09
Overall	**96.50**	**95.78**	**96.16**	**95.88**

Figure 4.7 shows per-client accuracy over 20 federated rounds, illustrating both the convergence behavior and the variation in learning dynamics across clients. Accuracy climbs quickly in the first few rounds (e.g., reaching 67% by round 2). This fast initial learning is an expected outcome of our design because the participation ratio is 100% and each round performs multiple local epochs on the clients' pseudo-labeled data. This allows clients to learn their local pseudo-labeled data thoroughly before each aggregation. Then, the plot visualizes the challenges of heterogeneity and



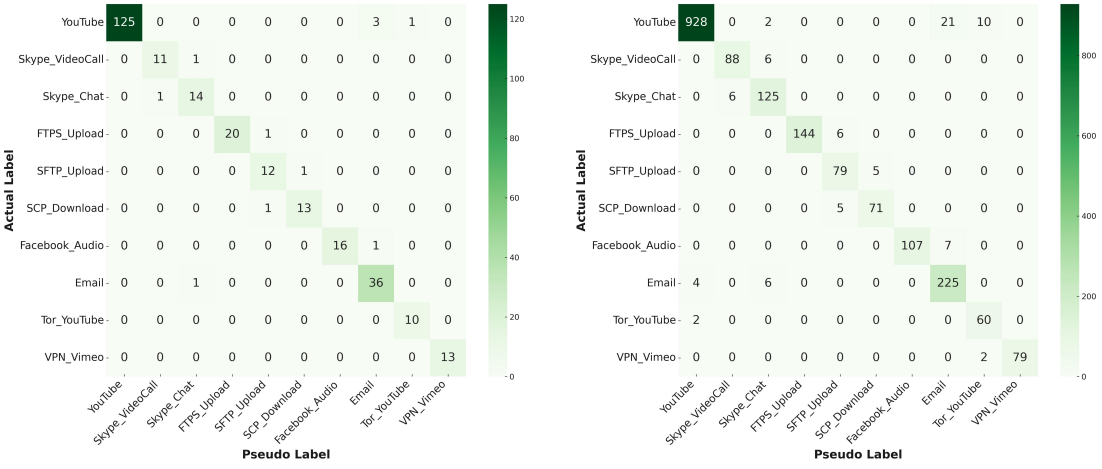
(a) Client 1.

(b) Client 2.



(c) Client 3.

(d) Client 4.



(e) Client 5.

(f) Overall.

Figure 4.5: Final classifier confusion matrices in FL: (a)-(e) clients 1-5, (f) overall.

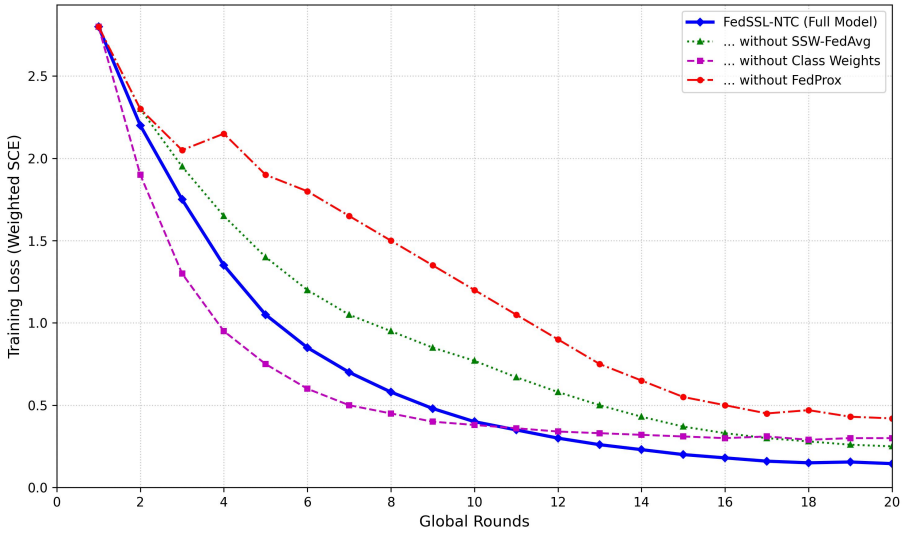


Figure 4.6: Training loss convergence of the final classifier under key ablation settings.

the effectiveness of our solution. Clients with smaller data volumes (e.g., client 5) exhibit faster early convergence but then stabilize smoothly like other clients, demonstrating robustness to quantity skew through SSW aggregation. The near-overlap of all five curves during 20 rounds reflects robustness to non-IID and class-imbalance conditions. Extending the plot to 20 rounds confirms that, beyond round 15, changes are marginal, consistent with the convergence indicated by Table 4.5.

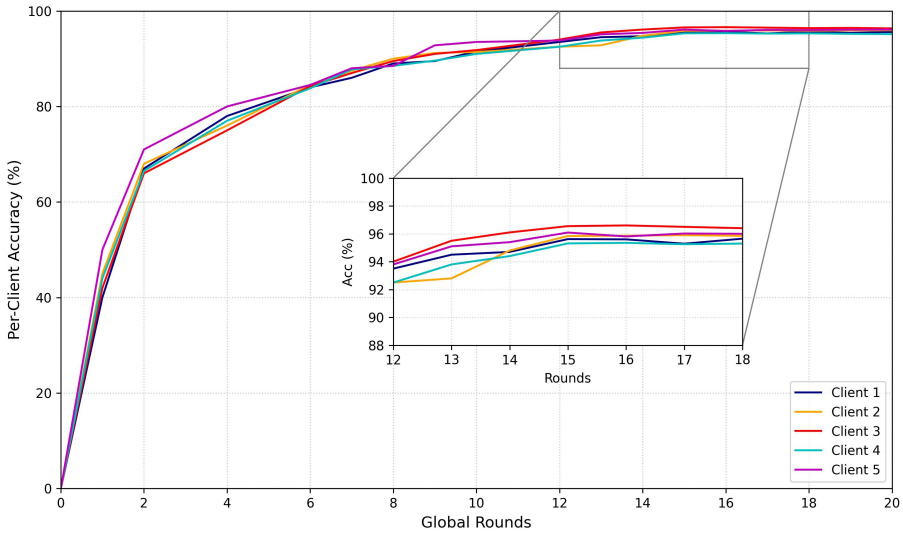


Figure 4.7: Per-client accuracy convergence across 20 federated rounds.

Table 4.8 summarizes the computational analysis for both phases: Phase I (AE + TabCL) and Phase II. On average, Phase I takes 27.8 s per local round at 18k flows/s with a modest 1.5 GB peak VRAM, indicating the SSL stage is the main compute sink (as expected due to TabCL's contrastive operations). Phase II is lightweight (≈8.8 s/round, ≈31.2k flows/s, 0.9 GB VRAM), making frequent global refreshes practical even under drift. This modest memory footprint confirms the models are lightweight enough for deployment on resource-constrained edge devices. Server aggregation is trivial (< 0.05 s/round), so wall-clock time is dominated by parallel client training, which favors near-linear scalability as clients increase or datasets grow. Overall, this analysis shows FedSSL-NTC is practical to deploy and likely to scale to larger, multi-client settings

Table 4.8: Computational profile of FedSSL-NTC (Self-gen+ISCX dataset).

Metric	**Phase I**	**Phase II**
Client-Side (avg. over 5 clients)		
Time per Local Round (s)	27.81	8.78
Training Throughput (flows/s)	18,243	31,148
Peak GPU Memory (GB)	1.50	0.90
Server-Side		
Aggregation Time per Round (s)	0.05	0.02

A comprehensive comparison of the centralized framework in chapter 3 and the novel federated learning approach is provided in Table 4.9. Despite the difficulties posed by non-IID data and class imbalance, the FedSSL-NTC maintains strong performance while enhancing data privacy and reducing centralized computational demands. Additionally, the federated configuration yields a 4–5× reduction in per-round training time across stages. This improvement stems from parallel client-side training on smaller local partitions, with communication/aggregation overheads remaining modest relative to the per-round computation.

Table 4.9: Performance metrics of SSL branches and the final classifier under Federated vs. Centralized training.

Metric	**SSL Autoencoder**		**SSL TabCL**		**SSL Fusion (AE+TabCL)**		**Final Classifier**	
	Federated	**Centralized**	**Federated**	**Centralized**	**Federated**	**Centralized**	**Federated**	**Centralized**
Accuracy (%)	86.82	87.17	88.62	89.22	89.30	89.91	95.88	96.29
Precision (%)	87.60	88.30	89.55	90.10	90.02	90.40	96.50	96.68
Recall (%)	86.16	86.50	88.12	88.60	88.64	89.10	95.78	96.31
F1-score (%)	86.84	87.20	88.82	88.90	89.32	89.30	96.16	96.41
Training time (Sec)	6.32	31.19	21.61	107.86	27.90	142.55	8.78	43.33

To benchmark the advancements of FedSSL-NTC, Table 4.10 contrasts its metrics with recent SOTA techniques. For fair and reproducible comparisons, we re-implemented (or faithfully followed public code/specifications for) all baselines under the same data protocol, client splits, metrics, and training budget, and tuned hyperparameters. Our federated model (95.88% accuracy) successfully closes the gap on our centralized works [55, 67, 93], achieving near-centralized performance while ensuring privacy. More importantly, it clearly outperforms all other federated baselines. This includes the FSSL methods, such as FLAECNN [87] (94.20%), FEAT [34] (93.05%), FS-GAN [89] (89.93%), and FCAL [90] (F1 = 94.75% and acc= 94.61%). It is important to note that while [87] and [90] reported higher accuracies (e.g., 96.1% and 97.7% respectively), they used the 5-class version of the ISCX dataset originally; our results reflect the more challenging 10-class problem. Furthermore, our framework shows a significant ≈6% accuracy lead over FS-GAN [89]. Finally, as the 'Privacy' column indicates, FedSSL-NTC and FEAT are the only approaches to implement privacy-enhancement mechanisms (e.g., via SecAgg), whereas other FL baselines only offer partial privacy by keeping data local. Overall, the table shows that FedSSL-NTC provides stronger accuracy/F1 while addressing all associated challenges and meeting stricter privacy requirements in the same evaluation setting, making our model a more comprehensive and adaptable solution for modern network traffic classification.

Table 4.10: Performance Metrics of FedSSL-NTC and state-of-the-art methods on the self-gen + ISCX dataset

Methods	**Authors & Year**	**Setting**	**Privacy**	**Precision**	**Recall**	**F1-score**	**Accuracy**
Multitask learning fusion	L. Liu et al. 2024 [55]	Centralized	✗	90.24	91.2	90.05	92.76
Self-supervised FlowPic	E. Horowicz et al. 2024 [67]	Centralized	✗	92.80	95.22	93.5	93.73
FEAT (FL heterogeneity focus)	Y. Guo et al. 2023 [34]	Federated	✓	91.54	93.83	93.52	93.05
FLAECNN (FSSL, AE+CNN)	Z. Wang et al. 2024 [87]	Federated	◐	94.14	94.20	94.08	94.20
FCAL (Async FSSL)	Y. Yan et al. 2025 [90]	Federated	◐	94.52	94.61	94.75	94.61
FS-GAN (federated GAN)	Y. Xiao et al. 2024 [89]	Federated	◐	89.9	88.99	90.24	89.93
Our centralized prior work	Ours (Chapter 3)	Centralized	✗	96.68	96.31	96.41	**96.29**
FedSSL-NTC (this Chapter)	Ours	Federated	✓	96.50	95.78	96.16	**95.88**

✓ = Addressed (e.g., via SecAgg, DP) ◐= Partially Addressed (basic FL) ✗ = Not Addressed

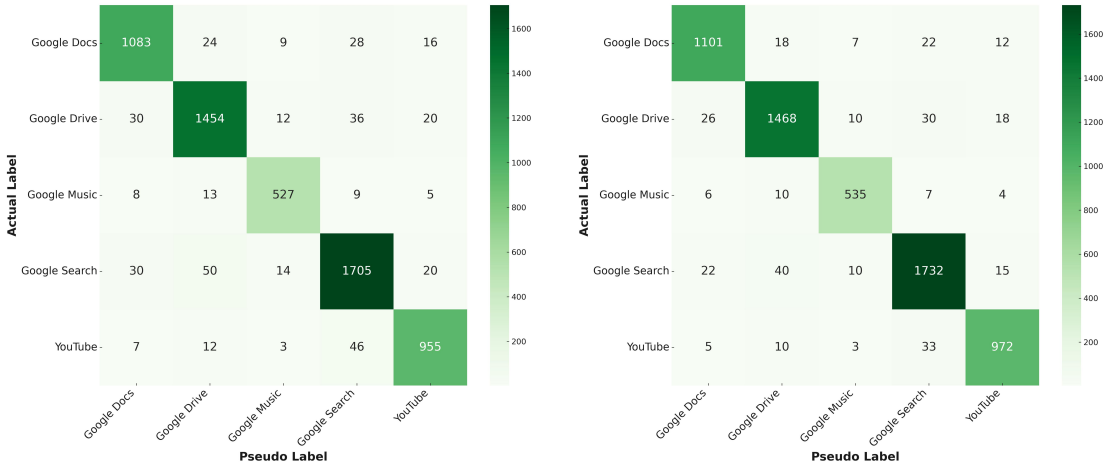
4.4.3 Results on UCDavis–QUIC Dataset

We repeat the two–stage evaluation on UCDavis–QUIC [39] to gauge generalization to modern encrypted traffic. Consistent with our data protocol, only a small, stratified labeled subset ($\approx 5\%$ per class) is retained centrally for head fine–tuning; the remaining flows are treated as unlabeled and partitioned across four clients with unequal sizes and skewed class composition. The core model architectures and optimizers remain consistent with those listed in Table 4.3. However, because the QUIC dataset is smaller and simpler (fewer flows and 5 classes vs. 10), the framework converges significantly faster. Therefore, we adjusted key training parameters: AE/TabCL local epochs 60, CL epochs 20, and final MLP epochs 30, and the total Communication Rounds R was lowered from 20 to 15 for both phases. All other settings are unchanged.

Clients run local SSL pretraining with FedProx, and the server aggregates with SSW FedAvg to respect client data volume. We report each branch (constraint–consistent AE and class–conditioned, constraint–preserving dual head TabCL) and a fusion that provides one pseudo–label per flow. Table 4.11 lists per–client and overall pseudo–label accuracy. The overall accuracies are 93.59% for AE, 94.97% for TabCL, and 95.55% for the fusion. Figure 4.8 shows the corresponding overall confusion matrices for AE and TabCL.

Table 4.11: Pseudo-label accuracy on QUIC under federated SSL.

Clients	**Accuracy (%)**		
	AE	**TabCL**	**Fusion (AE+TabCL)**
Client 1	93.45	94.78	95.38
Client 2	93.82	95.12	95.61
Client 3	93.73	95.08	95.71
Client 4	93.36	94.90	95.49
Overall	**93.59**	**94.97**	**95.55**



(a) AE pseudo-labels (overall).

(b) TabCL pseudo–labels (overall).

Figure 4.8: QUIC pseudo–label confusion matrices in FL: (a) AE, (b) TabCL.

Phase II trains the downstream MLP on the fusion pseudo–labeled pools. Because QUIC is smaller, we vary rounds up to 10. Table 4.12 shows the final classifier's accuracy versus rounds; we reach 98.24% at 15 rounds.

Table 4.12: Final classifier accuracy on QUIC at 15 rounds.

Rounds	**Accuracy (%)**
2	86.41
4	90.11
7	94.36
10	**98.24**
15	98.21

To provide a complete component analysis on the QUIC dataset, we performed a comprehensive ablation study mirroring the structure of the previous subsection. The results in Table 4.13 confirm our findings under this different non-IID setting. The data shows that the Fusion pseudo-labels provide the best performance, and that all FL components (SSW-FedAvg, FedProx, and Class Weights) are critical for overcoming the pathological label skew and achieving the final 98.24% accuracy.

Table 4.13: Ablation study on FedSSL-NTC using the QUIC dataset.

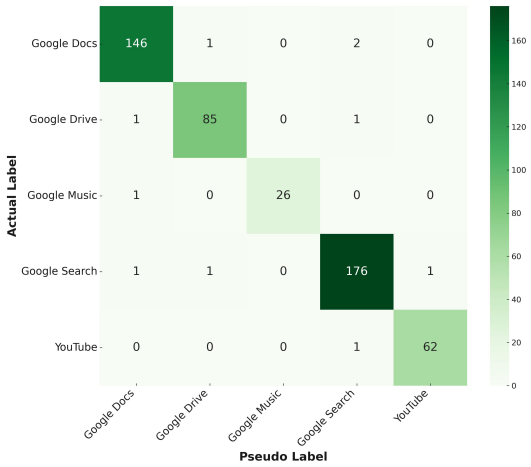
Setting	Macro-F1 (%)	Accuracy (%)
1. Pseudo-label Source (with Full Model)		
AE-based Pseudo-Labels	96.20	96.17
TabCL-based Pseudo-Labels	97.48	97.31
Fusion (AE $\oplus$ TabCL)	**98.34**	**98.24**
2. Incremental Build-up (on Fusion Labels)		
Simple FedAvg (Baseline)	81.92	82.68
+ SSW-FedAvg	85.28	85.95
+ FedProx	96.85	96.94
+ Class Weights	**98.34**	**98.24**
+ SecAgg (Full Model)	**98.34**	**98.24**
3. Subtractive Ablation (on Fusion Labels)		
Full Model (Ours)	**98.34**	**98.24**
... without Class Weights	96.85	96.94
... without FedProx	95.03	95.14
... without SSW-FedAvg	94.49	94.57
... without CL (Noisy Labels)	96.04	96.12

Figure 4.9 reports the final classifier confusion matrices for the four clients and the overall model; the matrices show uniformly low off–diagonal mass after CL weighting and FedProx stabilization. Table 4.14 summarizes precision/recall/F1/accuracy per client and overall; the aggregate accuracy is 98.24%.

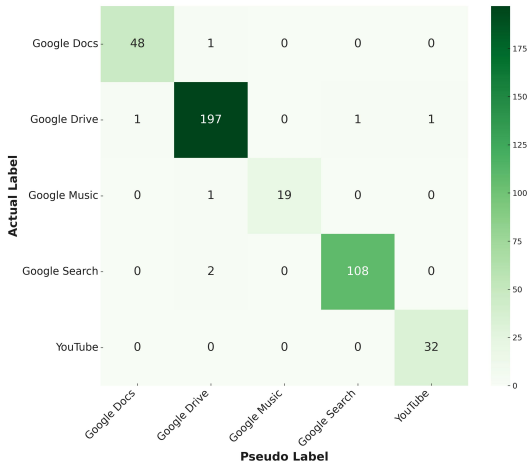
Table 4.14: Per-client Final classifier metrics on QUIC.

Client	Precision	Recall	F1–score	Accuracy
Client 1	98.35	97.92	98.13	98.05
Client 2	98.42	98.12	98.27	98.19
Client 3	98.77	98.43	98.60	98.58
Client 4	98.55	98.16	98.35	98.21
Overall	**98.52**	**98.16**	**98.34**	**98.24**

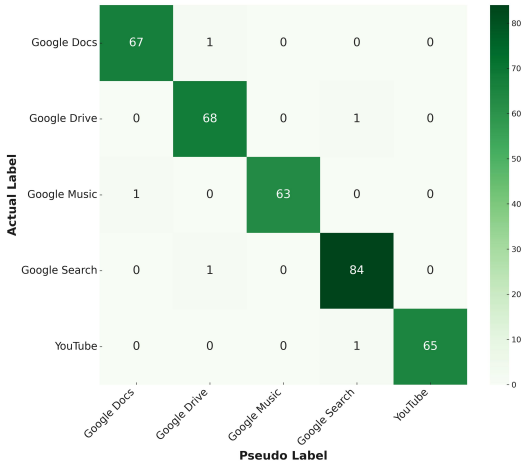
For completeness, Table 4.15 compares SSL branches and the final classifier under federated vs. centralized training on QUIC. Centralized pseudo–label accuracies (AE 94.07%, TabCL 95.65%, Fusion 96.11%) and final accuracy (98.76%) are in line with our revised centralized study; training times per epoch for the centralized AE and TabCL are ≈ 22.21 s and ≈ 51.95 s, respectively. In the federated configuration, parallel client updates compensate for communication overhead, yielding



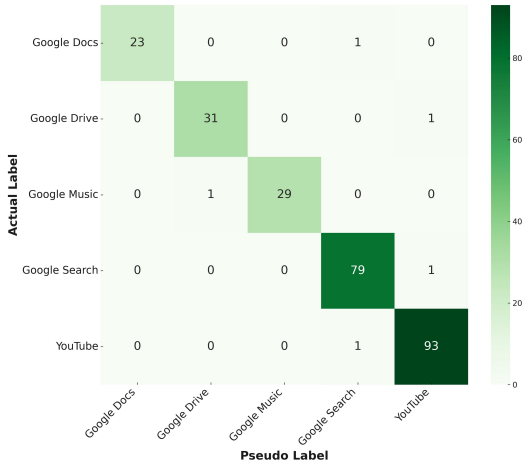
(a) Client 1.



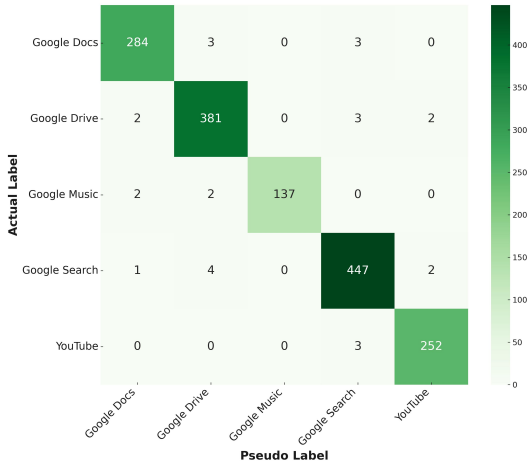
(b) Client 2.



(c) Client 3.



(d) Client 4.



(e) Overall.

Figure 4.9: Final classifier confusion matrices on QUIC: (a)–(d) clients 1–4, (e) overall.

materially shorter training times while attaining accuracy close to the centralized ceiling.

Table 4.15: SSL branches and final classifier under Federated vs. Centralized training (QUIC).

Metric	SSL Autoencoder		SSL TabCL		SSL Fusion (AE+TabCL)		Final Classifier	
	Federated	Centralized	Federated	Centralized	Federated	Centralized	Federated	Centralized
Accuracy (%)	93.59	94.07	94.97	95.65	95.55	96.11	98.24	98.76
Precision (%)	93.90	94.20	95.20	95.80	95.70	96.40	98.52	98.74
Recall (%)	93.40	93.90	94.70	95.40	95.40	95.80	98.16	98.69
F1–score (%)	93.65	94.05	94.95	95.60	95.55	96.10	98.34	98.71
Training time (Sec)	5.92	22.21	14.28	51.95	20.91	73.55	6.10	13.96

Table 4.16 contrasts FedSSL–NTC with strong QUIC baselines. The FlowPic SSL approach performs well in recall for encrypted traffic, while the prior centralized multi–task fusion also attains solid accuracy. The federated semi–supervised QUIC study (FSSL) reports 97.81% under an incremental client–sampling scheme, but it does not include CL–based noise mitigation or class–reweighting; our method reaches 98.24% in a strictly privacy-enhancing federated setting with explicit defenses for non–IID and class imbalance.

Table 4.16: Performance Metrics of FedSSL-NTC and state-of-the-art methods on QUIC dataset.

Methods	Authors & Year	Setting	Privacy	Precision	Recall	F1-score	Accuracy
Multitask learning fusion	L. Liu et al., 2024 [55]	Centralized	✗	94.20	94.70	94.45	94.67
Self-supervised FlowPic	E. Horowicz et al., 2024 [67]	Centralized	✗	98.10	98.45	98.27	98.30
FSSL (incremental sampling)	Y. Penge et al., 2021 [86]	Federated	◐	97.66	97.47	97.53	97.81
Our centralized prior work	Ours (Chapter 3)	Centralized	✗	98.74	98.69	98.71	**98.76**
FedSSL-NTC (this paper)	Ours	Federated	✓	98.52	98.16	98.34	**98.24**

✓ = Addressed (e.g., via SecAgg, DP) ◐= Partially Addressed (basic FL) ✗ = Not Addressed

4.5 Conclusion

This chapter presented FedSSL-NTC, a robust and privacy-aware federated framework that successfully adapts our centralized SSL and CL pipeline to distributed environments. The framework operates in two phases: collaborative SSL pretraining to generate local pseudo-labels, followed by client-side noise mitigation using traffic-adapted CL. To address the inherent challenges of federated networks, we integrated FedProx and class-weighted losses to stabilize training under non-IID and imbalanced distributions, while employing SSW FedAvg to manage quantity skew. Crucially, the integration of a tailored SecAgg protocol ensures that model updates remain cryptographically protected, providing a strong privacy guarantee beyond simple data locality.

The experimental evaluation demonstrates the framework's resilience, achieving 95.88% accuracy on the Self-generated/ISCX dataset and 98.24% on UCDavis-QUIC. These results confirm that FedSSL-NTC attains near-centralized performance while significantly reducing training time through parallel client updates. With the presentation of this federated solution, the technical contributions of this thesis are complete. The following chapter summarizes the overall conclusions of this research and outlines potential directions for future work.

Chapter 5

Conclusions and Future Work

This final chapter summarizes the thesis's key contributions and then discusses several promising directions for future research.

5.1 Conclusions

This thesis has addressed key challenges in NTC by developing innovative frameworks that leverage SSL and CL in both centralized and federated settings. As outlined in Chapter 1, NTC is essential for modern networks, enabling traffic engineering, QoS optimization, and security in environments dominated by encrypted and dynamic traffic. However, traditional methods fall short due to encryption, while supervised approaches suffer from label scarcity, and distributed data raises privacy and heterogeneity concerns.

In Chapter 3, we introduced a centralized framework that combines traffic-adapted SSL with CL to generate and refine high-quality pseudo-labels from unlabeled flows. By employing two complementary SSL branches—a constraint-consistent AE for reconstruction and an enhanced TabCL with dual-head temperatures and class-conditioned views—we achieved robust representations tailored to heterogeneous flow features. The traffic-adapted CL stage, with per-class quantile thresholds, calibration-aware weighting, and balanced retention, effectively mitigated pseudo-label noise. Evaluations on a self-generated unlabeled dataset combined with ISCX VPN-nonVPN, and the UC-Davis–QUIC dataset, demonstrated accuracies of 96.29% and 98.76%, respectively, outperforming

supervised and SSL baselines while using minimal labeled data. Building on this, Chapter 4 extended the approach to a federated context with FedSSL-NTC, enabling privacy-aware NTC without sharing raw traffic. Clients perform local SSL pretraining and CL-based denoising, followed by federated classifier training using FedProx for non-IID stability, class-weighted losses for imbalance, SSW FedAvg for fair aggregation, and a tailored SecAgg protocol to protect updates. Results showed accuracies of 95.88% on ISCX and 98.24% on QUIC, closely matching centralized performance with 4–5× faster training due to parallelism. Comparisons in Chapter 4 confirmed superiority over federated SSL baselines.

In summary, this thesis provides a complete and practical methodology for modern NTC. It begins by solving the core problem of supervised learning from unlabeled, noisy data and ultimately translates that solution into a secure, robust, and efficient federated framework suitable for real-world deployment.

5.2 Future work

Future research can extend this thesis in several directions to enhance practicality and robustness. For the centralized framework, incorporating online incremental learning could allow dynamic updates to representations and thresholds as traffic patterns evolve, addressing limitations in offline pretraining. Evaluating on more diverse datasets, including rare enterprise applications and emerging protocols like those in 6G networks, would improve generalization.

In the federated setting, communication-efficient variants, such as model compression or sparse updates, could reduce overhead in large-scale deployments with 50+ clients. Integrating stronger privacy mechanisms like differential privacy alongside SecAgg would provide formal guarantees against inference attacks. Adaptive client sampling under concept drift could further handle evolving distributions in IoT or 5G/6G environments. To further enhance generalization, the framework could be tested against a wider array of network traffic, including more diverse enterprise applications, IoT-specific protocols, and emerging encrypted protocols.

Bibliography

[1] A. Azab, M. Khasawneh, S. Alrabaee, K. K. R. Choo, and M. Sarsour, "Network traffic classification: Techniques, datasets, and challenges," *Digital Communications and Networks*, vol. 10, no. 3, pp. 676–692, Apr. 2024.

[2] R. Poorzare, D. N. Kanellopoulos, V. K. Sharma, P. Dalapati, and O. P. Waldhorst, "Network digital twin toward networking, telecommunications, and traffic engineering: A survey," *IEEE Access*, vol. 13, pp. 16 489–16 538, Jan. 2025.

[3] E. Paolini, L. Valcarenghi, L. Maggiani, and N. Andriolli, "Real-time network packet classification exploiting computer vision architectures," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1155–1166, Feb. 2024.

[4] A. S. Khan, M. A. Sattar, K. Nisar, A. A. Ibrahim, N. B. Annuar, J. b. Abdullah, and S. K. Memon, "A survey on 6g enabled light weight authentication protocol for uavs, security issues, open research issues and future research directions," *MDPI Applied Science Journal*, vol. 13, no. 1, Dec. 2023.

[5] C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, "6g internet of things: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 359–383, Jan. 2022.

[6] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "A comprehensive survey on 6g networks: Applications, core services, enabling technologies, and challenges," *arXiv:2101.12475*, Jan. 2021.

[7] M. Loy, "How much data is generated per day?" *Digital Silk*, Jun. 2025.

[8] M. Saqib, H. Elbiaze, and R. H. Glitho, "Adaptive in-network traffic classifier: Bridging the gap for improved qos by minimizing misclassification," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 677–689, Jan. 2024.

[9] Y. Feng, J. Li, J. Mirkovic, C. Wu, C. Wang, H. Ren, J. Xu, and Y. Liu, "Unmasking the internet: A survey of fine-grained network traffic analysis," *IEEE Communications Surveys & Tutorials*, Feb. 2025.

[10] M. Abbasi, A. Shahraki, and A. Tasherkordi, "Deep learning for network traffic monitoring and analysis (ntma): A survey," *Computer Communications*, vol. 170, pp. 19–33, Mar. 2021.

[11] D. D. Monda, A. Montieri, V. Persico, P. Voria, M. D. Ieso, and A. Pescapè, "Few-shot class-incremental learning for network intrusion detection systems," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 6736–6757, Oct. 2024.

[12] A. Blika, S. Palmos, G. Doukas, V. Lamprou, S. Pelekis, M. Kontoulis, C. Ntanos, and D. Askounis, "Federated learning for enhanced cybersecurity and trustworthiness in 5g and 6g networks: A comprehensive survey," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 3094–3130, Aug. 2025.

[13] R. Raveendran and R. Menon, "An efficient method for internet traffic classification and identification using statistical features," *Int. Journal Eng. Res. Tech.*, vol. 4, no. 7, pp. 297–303, 2015.

[14] G. Draper-Gil, A. Habibi, M. Saiful, and A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," *Proc. 2nd Int. Conf. Inf. Syst. Security Privacy (ICISS)*, pp. 407–414, 2016.

[15] A. Sharma and A. H. Lashkari, "A survey on encrypted network traffic: A comprehensive survey of identification/classification techniques, challenges, and future directions," *Computer Networks*, vol. 257, pp. 110 984–111 012, Feb. 2025.

[16] P. Khandait, N. Hubballi, and B. Mazumdar, "Efficient keyword matching for deep packet inspection based network traffic classification," *IEEE International Conference on Communication Systems & Networks (COMSNETS)*, pp. 567–570, Mar. 2020.

[17] A. Nascita, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapè, "A survey on explainable artificial intelligence for internet traffic classification and prediction, and intrusion detection," *IEEE Communications Surveys & Tutorials*, Nov. 2024.

[18] Y. S. Razooqi and A. Pekar, "Vpn traffic analysis: A survey on detection and application identification," *IEEE Access*, vol. 13, pp. 132 830–132 848, Jul. 2025.

[19] J. Krupski, M. Iwanowski, and W. Graniszewski, "On the right choice of data from popular datasets for internet traffic classification," *Computer Communications Journal*, vol. 233, p. 108068, Mar. 2025.

[20] Y. Heng, V. Chandrasekhar, and J. G. Andrews, "Utmobilenettraffic2021: A labeled public network traffic dataset," *IEEE Networking Letters*, vol. 3, no. 3, pp. 156–160, 2021.

[21] J. L. Guerra, C. Catania, and E. Veas, "Datasets are not enough: Challenges in labeling network traffic," *Comput. Secur.*, vol. 120, p. 102810, Jun. 2022.

[22] D. Soukup, P. Tisovčík, K. Hynek, and T. Čejka, "Towards evaluating quality of datasets for network traffic domain," *IEEE Int. Conf. Network and Service Management (CNSM)*, 2021.

[23] Z. Wang, Y. Yang, and Y. Wang, "A survey of encrypted traffic classification: Datasets, representation, approaches and future thinking," *IEEE International Conference on Computer and Information Science (ICIS)*, Dec. 2024.

[24] K. Lin, X. Xu, and Y. Jiang, "A new semi-supervised approach for network encrypted traffic clustering and classification," *IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, May 2022.

[25] Z. Guo, B. Lu, W. Wu, M. Guo, Z. Liu, and X. Lin, "A semi-supervised learning framework for encrypted traffic classification based on supervised contrastive learning and masked sequence

prediction tasks," *Proc. IEEE International Conference on Advanced Algorithms and Control Engineering (ICAACE)*, 2025.

[26] M. S. Towhid and N. Shahriar, "Encrypted network traffic classification using self-supervised learning," *Proc. IEEE International Conference on Network Softwarization (NetSoft)*, Aug. 2022.

[27] H. Mun and Y. Lee, "Internet traffic classification with federated learning," *MDPI Electronics Journal*, vol. 10, no. 1, 2021.

[28] H. A. Tran, H. T. T. Binh, and A. Mellouk, "Federated learning for network traffic classification: A knowledge consolidation approach," *IEEE Transactions on Network Science and Engineering*, Jul. 2025.

[29] L. Liu, Y. Tian, C. Chakraborty, J. Feng, Q. Pei, L. Zhen, and K. Yu, "Multilevel federated learning-based intelligent traffic flow forecasting for transportation network management," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1446–1458, May 2023.

[30] J. Kone, H. B. McMahan, F. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[31] N. Truong, K. Sun, S. Wang, F. Guitton, and Y. Guo, "Privacy preservation in federated learning: An insightful survey from the gdpr perspective," *Computers & Security*, vol. 110, p. 102402, 2021.

[32] C. Dwork, "Differential privacy," *International Colloquium on Automata, Languages, and Programming*, pp. 1–12, 2006.

[33] K. Bonawitz, V. Ivanov, B. Kreuter, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[34] Y. Guo and D. Wang, "Feat: A federated approach for privacy preserving network traffic classification in heterogeneous environments," *IEEE Internet of Things Journal*, vol. 10, no. 2, pp. 1274–1285, Jan. 2023.

[35] D. M. Jimenez-Gutierrez, M. Hassanzadeh, A. Anagnostopoulos, I. Chatzigiannakis, and A. Vitaletti, "A thorough assessment of the non-iid data impact in federated learning," *arXiv:2503.17070*, Jul. 2025.

[36] W. Cui, R. Hosseinzadeh, J. Ma, T. Wu, Y. Sui, and K. Golestan, "Tabular data contrastive learning via class-conditioned and feature-correlation based augmentation," *arXiv:2404.17489*, Apr. 2024.

[37] C. G. Northcutt, L. Jiang, and I. L. Chuang, "Confident learning: Estimating uncertainty in dataset labels," *Journal of Artificial Intelligence Research*, vol. 70, pp. 1373–1411, Apr. 2021.

[38] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Fedprox: Federated optimization in heterogeneous networks," *arXiv:1812.06127*, Apr. 2020.

[39] S. Rezaei and X. Liu, "How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets," *arXiv:1812.09761*, May 2020.

[40] N. Hubballi, M. Swarnkar, and M. Conti, "Bitprob: Probabilistic bit signatures for accurate application identification," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1730–1741, Sep. 2020.

[41] R. H. Serag, M. S. Abdalzaher, H. Elsayed, M. Sobh, M. Krichen, and M. Salim, "Machine-learning-based traffic classification in software-defined networks," *Electronics*, vol. 13, no. 6, Mar. 2024.

[42] A. Salau and M. Beyene, "Software defined networking based network traffic classification using machine learning techniques," *Scientific Reports*, vol. 14, Aug. 2024.

[43] M. R. Choudhury, P. Acharjee, and A. T. George, "Network traffic classification using supervised learning algorithms," *IEEE International Conference on Computer, Electrical & Communication Engineering (ICCECE)*, Apr. 2023.

[44] W. A. Aziz, H. K. Qureshi, A. Iqbal, A. Al-Dulaimi, and S. Al-Rubaye, "Towards accurate categorization of network ip traffic using deep packet inspection and machine learning," *IEEE Global Communications Conference (GLOBECOM)*, Dec. 2023.

[45] J. Gómez, V. H. R. no, and G. Ramirez-Gonzalez, "Traffic classification in ip networks through machine learning techniques in final systems," *IEEE Access*, vol. 11, pp. 44 932–44 940, May 2023.

[46] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature Publishing Group UK*, vol. 521, no. 7553, pp. 436–444, May 2015.

[47] G. D'Angelo and F. Palmieri, "Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial–temporal features extraction," *Journal of Network and Computer Applications*, vol. 173, p. 102890, Jan. 2021.

[48] A. Jenefa, S. Sam, V. Nair, B. G. Thomas, A. S. George, and R. Thomas, "A robust deep learning-based approach for network traffic classification using cnns and rnns," *IEEE International Conference on Signal Processing and Communication (ICSPC)*, pp. 106–111, May 2023.

[49] W. Lin and Y. Chen, "Robust network traffic classification based on information bottleneck neural network," *IEEE Access*, vol. 12, pp. 150 169–150 179, Oct. 2024.

[50] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[51] Y. Ma, Z. Li, H. Xue, and J. Chang, "A balanced supervised contrastive learning-based method for encrypted network traffic classification," *Computers & Security Journal*, vol. 145, p. 104023, Oct. 2024.

[52] H. Zhang, L. Yu, X. Xiao, Q. Li, F. Mercaldo, X. Luo, and Q. Liu, "Tfe-gnn: A temporal fusion encoder using graph neural networks for fine-grained encrypted traffic classification," *Proceedings of the ACM Web Conference*, pp. 2066–2075, Jul. 2023.

[53] F. Hu, S. Zhang, X. Lin, L. Wu, N. Liao, and Y. Song, "Network traffic classification model based on attention mechanism and spatiotemporal features," *EURASIP Journal on Information Security*, no. 6, Jul. 2023.

[54] X. Xiao, S. Wang, G. Hu, X. Luo, Q. Li, K. Mao, B. Zhang, and S. Xia, "Rbljan: Robust byte-label joint attention network for network traffic classification," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 3, pp. 2161–2178, Oct. 2024.

[55] L. Liu, Y. Yu, Y. Wu, Z. Hui, and J. Hu, "Method for multi-task learning fusion network traffic classification to address small sample labels," *Scientific Reports*, vol. 14, no. 1, p. 2518, Jan. 2024.

[56] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong, and A. A. Ghorbani, "Towards the development of a realistic multidimensional iot profiling dataset," *19th Annual International Conference on Privacy, Security & Trust (PST2022)*, Aug. 2022.

[57] R. Yang, A. Yu, L. Cai, and D. Meng, "Subspace clustering via graph auto-encoder network for unknown encrypted traffic recognition," *Cybersecurity*, vol. 5, no. 29, 2022.

[58] B. Gao, Y. Yang, Z. Gao, P. Yu, R. Lyu, and S. Chen, "Unsupervised network traffic classification based on multi-source synergistic distribution alignment," *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, Feb. 2024.

[59] W. Zhang, L. Zhang, X. Zhang, Y. Wang, P. Liu, and G. Gui, "Intelligent unsupervised network traffic classification method using adversarial training and deep clustering for secure internet of things," *MDPI Future Internet Journal*, vol. 15, no. 9, pp. 298–318, 2023.

[60] G. Jie, C. Tuo, Z. Jing, C. Qiong, S. Zhenan, L. Hao, and T. Dacheng, "A survey on self-supervised learning: Algorithms, applications, and future trends," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 9052–9071, 2024.

[61] M. Ishan and M. L. van der, "Self-supervised learning of pretext-invariant representations," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[62] L. Xiao, Z. Fanjin, H. Zhenyu, M. Li, Z. Wang, Z. Jing, T. J. Xiao, Z. Fanjin, Z. Hou, M. Li, Z. Wang, Z. Jing, and T. Jie, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 857–876, June 2023.

[63] J. Ashish, B. A. Ramesh, Z. M. Zaki, B. Debapriya, and F. Makedon, "A survey on contrastive self-supervised learning," *MDPI*, vol. 9, no. 1, Dec. 2020.

[64] H. Zhenyu, L. Xiao, C. Yukuo, D. Yuxiao, H. Yang, W. Chunjie, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," *Association for Computing Machinery*, vol. 22, no. 11, pp. 594—604, Aug. 2022.

[65] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," *Proceedings of the ACM Web Conference*, pp. 633–642, Feb. 2022.

[66] R. Zhao, M. Zhan, X. Deng, Y. Wang, Y. Wang, G. Gui, and Z. Xue, "Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, pp. 5420–5427, 2023.

[67] E. Horowicz, T. Shapira, and Y. Shavitt, "Self-supervised traffic classification: Flow embedding and few-shot solutions," *IEEE Transactions on Network and Service Management*, vol. 21, no. 3, pp. 3054–3067, Jun. 2024.

[68] Y. Huo, C. Song, M. Zhou, R. Lv, and Y. Yang, "A novel approach for semi-supervised network traffic classification," *IEEE International Conference on Advanced Infocomm Technology (ICAIT)*, pp. 64–70, Aug. 2022.

[69] S. Xu, J. Han, Y. Liu, H. Liu, and Y. Bai, "Few-shot traffic classification based on autoencoder and deep graph convolutional networks," *Scientific Reports*, vol. 15, Mar. 2025.

[70] M. A. Aleisa, "Traffic classification in sdn-based iot network using two-level fused network with self-adaptive manta ray foraging," *Scientific Reports*, vol. 15, Jan. 2025.

[71] A. R. Bahlali, A. Bachir, and A. Labed, "Self-supervised learning meets custom autoencoder classifier: A semi-supervised approach for encrypted traffic anomaly detection," *IEEE Access*, vol. 13, pp. 139 141–139 154, Aug. 2025.

[72] Y. Yuan, Y. Huang, X. Zeng, H. Mei, and G. Cheng, "M3s-upd: Efficient multi-stage self-supervised learning for fine-grained encrypted traffic classification with unknown pattern discovery," *Scientific Reports*, May 2025.

[73] S. Sattar, S. Khan, M. I. Khan, A. Akhmediyarova, O. Mamyrbayev, D. Kassymova, D. Oralbekova, and J. Alimkulova, "Anomaly detection in encrypted network traffic using self-supervised learning," *Scientific Reports*, vol. 15, Jul. 2025.

[74] X. Chen, M. Ding, X. Wang, Y. Xin, S. Mo, Y. Wang, S. Han, P. Luo, G. Zeng, and J. Wang, "ontext autoencoder for self-supervised representation learning," *International Journal of Computer Vision*, vol. 132, no. 1, pp. 208–223, 2024.

[75] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial intelligence and statistics*, pp. 1273–1282, 2017.

[76] W. Wen, Y. Li, and F. C. M. Lau, "Byzantine-resilient online federated learning with applications to network traffic classification," *IEEE Network*, vol. 37, no. 4, pp. 145–152, Oct. 2023.

[77] C. Qiu, "A network traffic classification method based on federated learning and extreme learning machine," *Proc. IEEE Int. Conf. Control, Electron. Comput. Technol. (ICCECT)*, pp. 284–289, 2023.

[78] Z. Jin, K. Duan, C. Chen, M. He, S. Jiang, and H. Xue, "Fedetc: Encrypted traffic classification based on federated learning," *Heliyon Journal*, vol. 10, no. 16, Aug. 2024.

[79] C. Qiao, M. Li, Y. Liu, and Z. Tian, "Transitioning from federated learning to quantum federated learning in internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 1, pp. 509–545, May 2024.

[80] Y. Li, Z. Guo, N. Yang, H. Chen, D. Yuan, and W. Ding, "Threats and defenses in the federated learning life cycle: A comprehensive survey and challenges," *IEEE Transactions on Neural Networks and Learning Systems*, May 2025.

[81] S. Zongxuan, R. Huo, S. Chuang, S. Wang, T. Huang, and F. R. Yu, "When communication networks meet federated learning for intelligence interconnecting: A comprehensive survey and future perspective," *IEEE China Communications*, vol. 22, no. 7, pp. 74–94, Aug. 2025.

[82] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54 024–54 033, 2019.

[83] R. Carillo, F. Cerasuolo, G. Bovenzi, and A. Pescapè, "Explainable federated class incremental learning for encrypted network traffic classification," *Computer Networks*, p. 111448, 2025.

[84] S. Yin, H. Li, A. Laghari, and A. Almadhor, "Flsn-mvo: edge computing and privacy protection based on federated learning siamese network with multi-verse optimization algorithm for industry 5.0," *IEEE Open Journal of the Communications Society*, 2024.

[85] L. Teng, Q. Feng, W. Zhao, and D. He, "Secure federated distillation framework for encrypted traffic classification," *International Conference on Information Security Practice and Experience*, pp. 1–19, 2024.

[86] Y. Peng, M. He, and Y. Wang, "A federated semi-supervised learning approach for network traffic classification," *International Journal of Network Management*, vol. 33, no. 3, Jan 2023.

[87] Z. Wang, Z. Li, M. Fu, Y. Ye, and P. Wang, "Network traffic classification based on federated semi-supervised learning," *Journal of Systems Architecture*, vol. 149, p. 103091, 2024.

[88] T. Qin, G. Cheng, Z. Yin, Y. Wei, Z. Yao, and Z. Chen, "Building robust traffic classifier under low quality data: a federated contrastive learning approach," *Digital Communications and Networks Journal*, Jun. 2025.

[89] Y. Xiao, R. Zhang, Y. Lou, and J. Fan, "Distributed traffic synthesis and classification in edge networks: A federated self-supervised learning approach," *IEEE Transactions on Mobile Computing*, vol. 22, no. 4, pp. 2134–2147, Apr. 2023.

[90] Y. Yan, Q. Yuan, W. Niu, and Y. Wang, "Fcal: An asynchronous federated contrastive semi-supervised learning approach for network traffic classification," *International Conference on Information and Communications Security*, pp. 419–437, 2025.

[91] C. Lin, C. Tseng, and L. An, "Employing federated semi-supervised learning for network traffic classification," *Journal of Network and Systems Management*, vol. 33, no. 3, p. 53, 2025.

[92] P. Wang, Z. Li, M. Fu, and M. Liu, "Fededge ai-tc: A semi-supervised traffic classification method based on trusted federated deep learning for mobile edge computing," *arXiv preprint arXiv:2308.06924*, 2023.

[93] E. Eslami and W. Hamouda, "Network traffic classification using self-supervised learning and confident learning," *IEEE Open Journal of the Communications Society (OJCOMS)*, vol. 6, pp. 9100–9120, Oct. 2025.

[94] E. Eslamii and W. Hamouda, "Fedssl-ntc: A robust federated self-supervised learning framework for network traffic classification under privacy constraints," *IEEE Open Journal of the Communications Society (OJCOMS)*, vol. 6, Nov. 2025.