PhD Interview Talk

Research Interests in Cloud–Edge AI

Jan 20, 2026

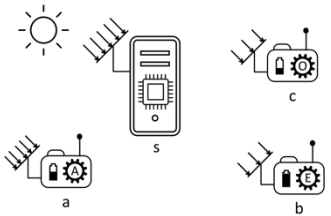
PhD Applicant: Junfei Zhan
zjf2024@seas.upenn.edu

Penn Engineering

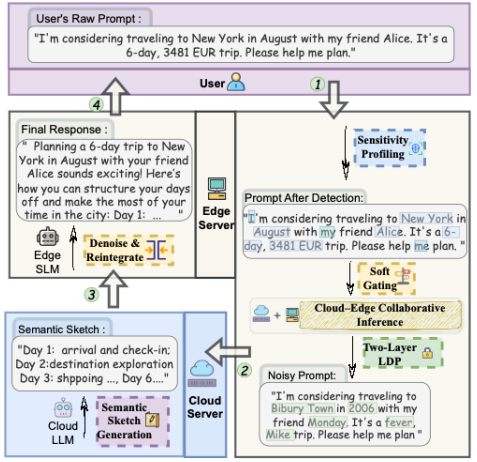
Academic Background

- Educational Background
  - University of Pennsylvania, MS in Electrical Engineering (GPA: 3.96/4.00)
  - University of Birmingham, BSc in Applied Math (First Class Honors)
  - Jinan University, BSc in Information & Computing Science (90.1/100)

- Research Experience

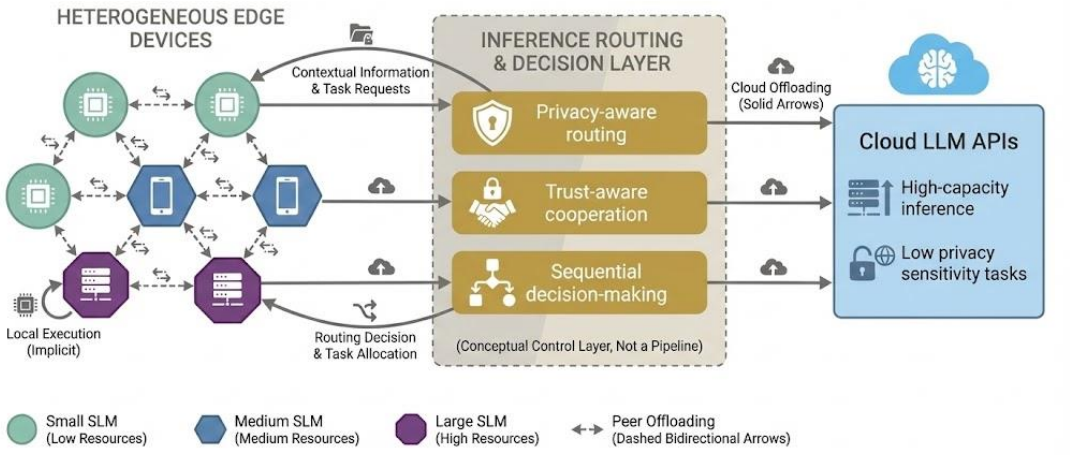


Green IoT networks



Cloud-Edge LLM system

PhD Goals

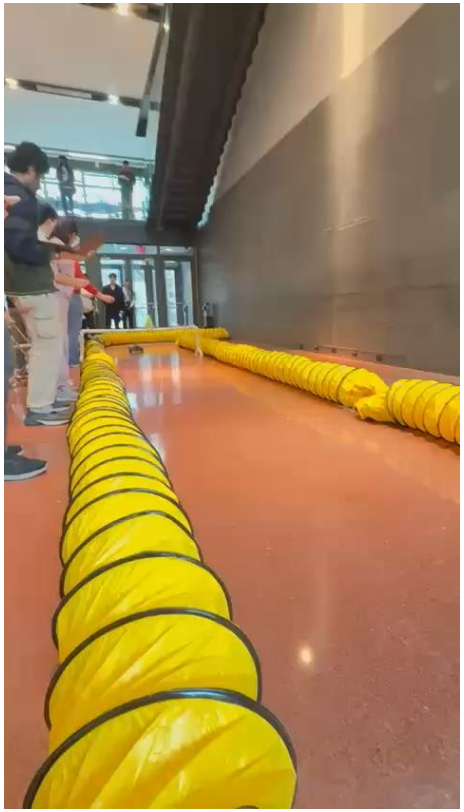


My research goal is to understand and design the interaction between AI inference and networked systems, with an emphasis on distributed and resource-constrained environments.

Ultimately, my goal is to develop into an independent researcher capable of identifying impactful research questions and formulating principled solutions in this field.

Technical Expertise

- Modeling & Optimization
  - Mixed-integer linear programming, optimal control and system-level modeling (MATLAB, Python)
- Machine Learning & Inference
  - PyTorch, scikit-learn, and evaluation of inference pipelines (Python)
- Systems & Performance Awareness
  - Linux-based system operation and performance-aware experimentation on edge platforms (e.g., NVIDIA Jetson Orin, F1TENTH)



Example: real-time inference and control on an embedded edge platform F1TENTH.

Motivation

Where Should LLMs Models Be Deployed?

Deployment Paradigm	Advantages	Disadvantages
Cloud-Centric	Enough computational resources; convenient model updates and maintenance.	High communication overhead, poor real-time response, and significant privacy risks.
Edge-Centric	Data remains local, offering better privacy protection and faster response times.	Limited on-device computation and memory, making it difficult to run large-scale models.

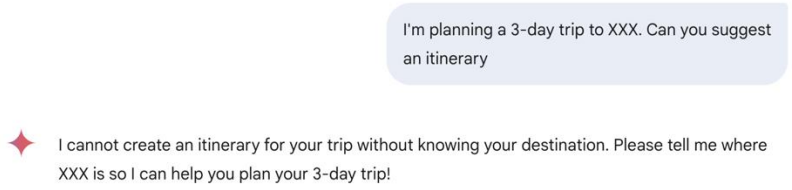
- User prompts often contain rich semantic context, including **location privacy** (e.g., home address, workplace), **identity-related privacy** (e.g., names, ID numbers. Cloud-centric deployment typically requires uploading the **entire prompt**, exacerbating privacy risks.
- User application scenarios require both the **intelligence and capacity of the cloud** and the **security and low latency of edge devices**, making privacy protection a critical concern.

Cloud-Edge Collaborative Inference

Motivation

Limitations of Existing Cloud–Edge Collaborative Privacy-Preserving Inference

- **Coarse-grained decision making:** User Existing methods rely on binary cloud/edge decisions based on simple risk estimation, lacking fine-grained semantic awareness of prompt context. This can lead to privacy leakage or unnecessary performance degradation. Representative works: CE-CoLLM (Jin and Wu, 2024) and CE-LSLM (Zhu and Yang, 2025).
- **Semantic distortion from perturbation:** Noise-based privacy mechanisms often corrupt sensitive tokens, distorting user intent and resulting in incomplete, ambiguous, or low-quality cloud-side inference. Representative works: Split-and-Denoise (Mai et al., 2024) and DP-Forward (Du et al., 2023).



Prism Framework

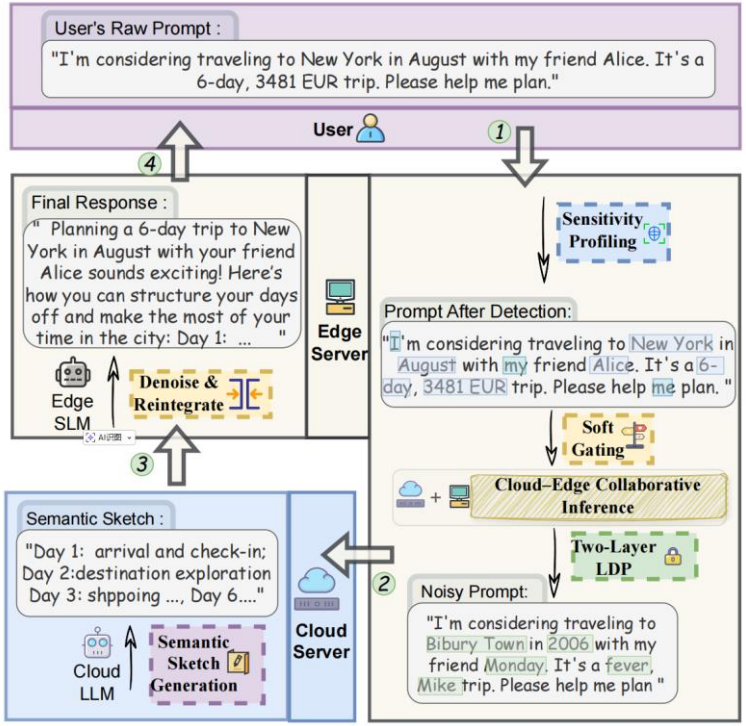
PRISM Architecture

PRISM (Full name): Privacy-aware Routing for Inference with Semantic Modulation

Core idea: Dynamically select inference routes based on semantic sensitivity and contextual risk.

Main Components

- Sensitivity-aware semantic analysis and Edge-side analysis of semantic content and privacy risk.
- Soft Routing decision: The edge determines whether inference should be performed on the cloud, edge, or collaboratively.
- Adaptive Local Differential Privacy (LDP): Privacy-preserving perturbation under collaborative inference settings.
- Semantic sketch collaboration: High-level semantic sketch generated in the cloud, with edge-side refinement and completion.



[1] Junfei Zhan, Haoxun Shen, Zheng Lin, and Tengjiao He. Prism: Privacy-aware routing for adaptive cloud–edge llm inference. Proceedings of the AAAI Conference on Artificial Intelligence, 2026.

Prism Framework

Sensitivity Profiling

Objective: To assess the presence of sensitive information in user prompts at the edge and compute a quantitative risk score and sensitivity profile that characterizes the overall privacy risk of the interaction.

Key Challenges:

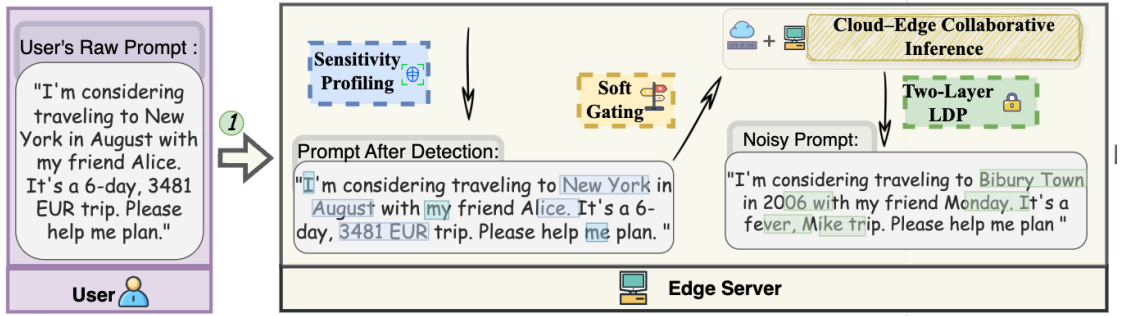
- Sensitivity profiling must be performed on resource-limited edge devices (e.g., smartphones, laptops) and completed before routing decisions, requiring lightweight models.
- Context-dependent sensitivity: The privacy risk of a token is highly context-dependent. For example, “Tokyo” is low-risk in “What is the capital of Japan?”, but it becomes sensitive in “I plan to travel from Tokyo next week”.
- Beyond binary classification: Existing methods often reduce privacy assessment to binary labels (sensitive vs. non-sensitive). In contrast, this work outputs a continuous sensitivity score and a multi-level risk profile, enabling more flexible and fine-grained routing decisions.

Prism Framework

Sensitivity Profiling – Pipeline of Calculating Risk Scores of user's prompts

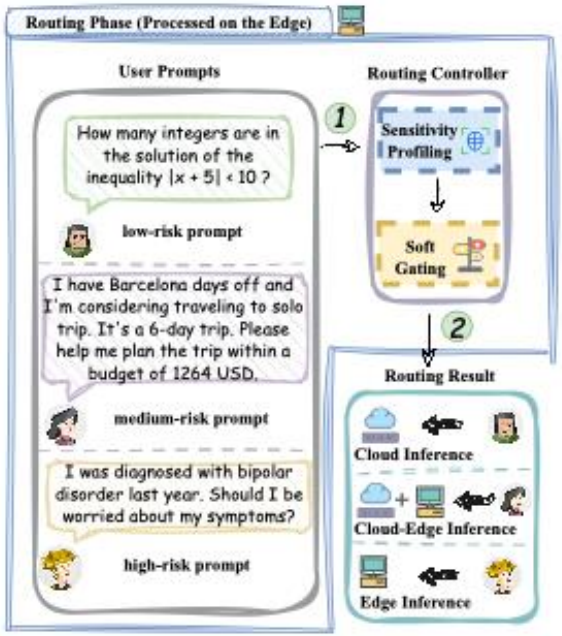
Main Steps:

- *Step 1:* Named Entity Recognition (NER): Given a prompt composed of multiple tokens, the system extracts a set of named entities to identify key semantic elements.
- *Step 2:* Entity Classification and Weight Assignment: Each entity is associated with a predefined category label (e.g., person, location, disease), and a predefined sensitivity weight is assigned based on its category.
- *Step 3:* The presence of first-person references or personal identifiers indicates user involvement, elevating the overall prompt sensitivity and causing all entities to be treated as potentially sensitive.
- *Step 4:* Context-Aware Risk Scoring: The system adjusts entity-level sensitivity scores based on the prompt context.



Prism Framework

Soft Gating



Key Idea

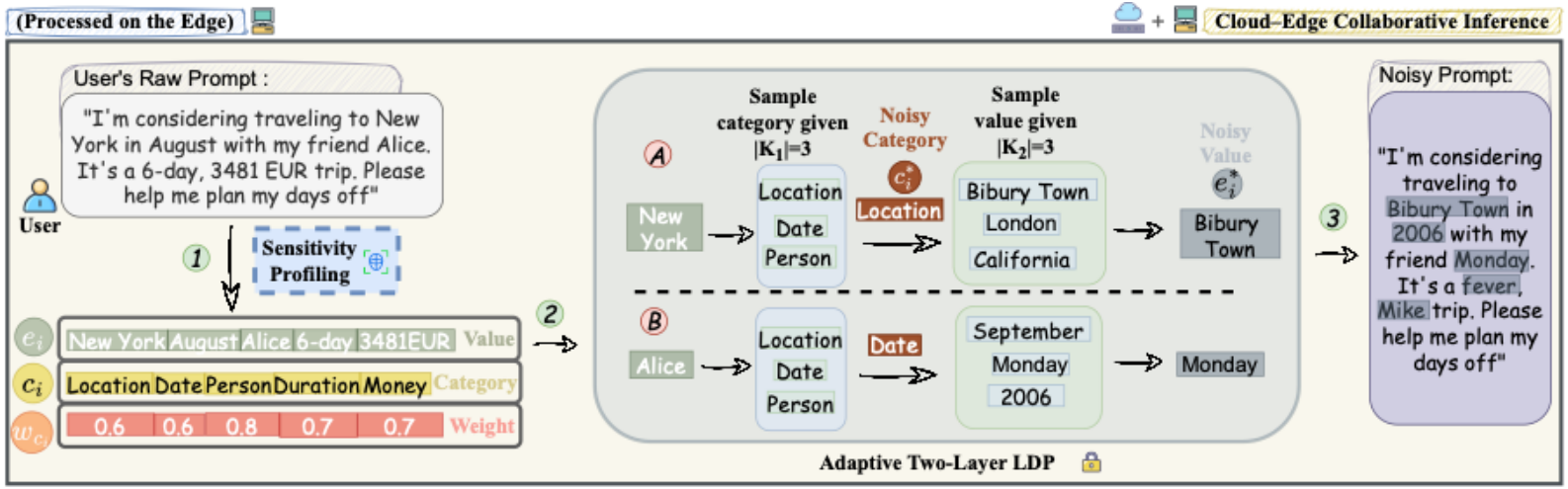
- Instead of a hard cloud/edge decision, we use a **soft gating mechanism** that maps **sensitivity signals** to a **probability distribution** over routing paths.
- This enables **context-aware and nuanced routing**, especially for ambiguous prompts.

How It Works

- **Input:** Sensitivity features derived at the edge (risk score + sensitivity mask).
- **Soft Routing:** A lightweight gating module outputs routing probabilities *(cloud / collaboration / local)* via a softmax layer.
- **Entropy Regularization:** An entropy penalty encourages **confident decisions** when appropriate, while retaining flexibility for uncertain cases.

Prism Framework

Adaptive Two-layer LDP



Prism Framework

Cloud–Edge Semantic Sketch Collaboration

Privacy-aware routing: Prompts selected for collaboration are first perturbed using **adaptive two-layer LDP**, producing a noisy prompt P^*.

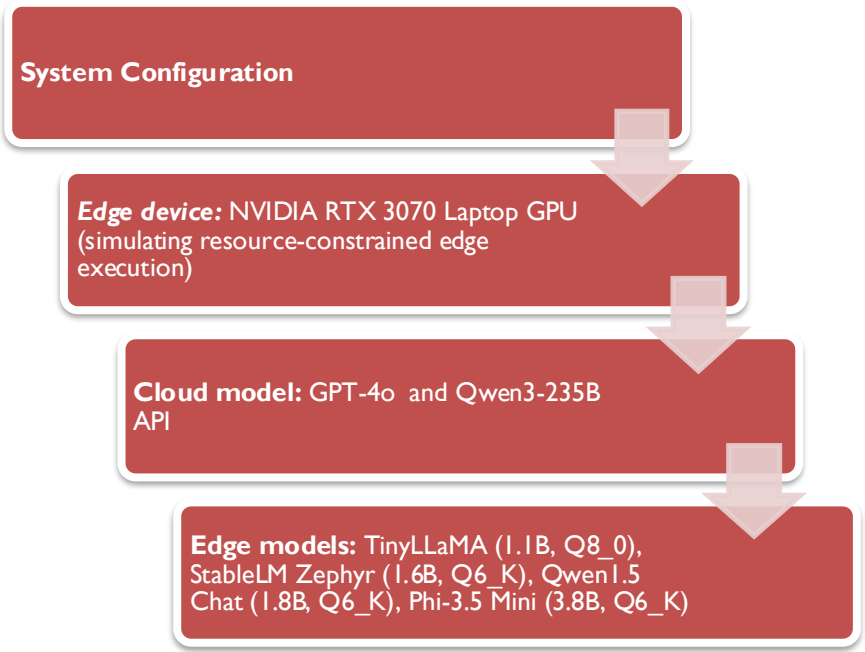
Cloud-side sketch generation: The cloud LLM receives only P^* (plain text, no embeddings).

Using few-shot prompting, it generates a **semantic sketch** S: concise, structured, free of sensitive entities

Edge-side reconstruction

The edge SLM combines the **original prompt** P (kept locally) with the sketch S to get the final response.

Experiments Setup



Evaluation Metrics

(1) Inference Quality (IQ): Rated on a 1–10 scale by GPT-4o, evaluating responses quality.

(2) Energy Consumption (EC): Total energy used during inference, measured in joules (J).

(3) Completion Time (Ct): End-to-end latency from prompt input to final response.

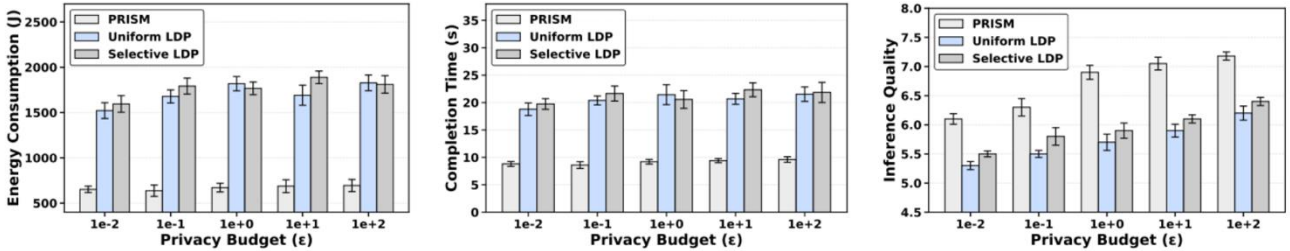
Dataset

- Constructed a balanced, controlled prompt dataset spanning four domains: Travel Planning, Medical Consultation, Banking Services, and General Knowledge *(non-sensitive)*.
- Each domain contains 40 prompts with diverse phrasing styles and entity distributions, designed to simulate realistic user–LLM interactions.

Evaluation

Method	Ct.(s)	Ec.(J)	IQ.
PRISM	7.92	687.16	6.88
Uniform LDP	20.56	1707.6	5.72
Selective LDP	21.22	1770.8	5.94
Edge-Only	17.84	1573.9	5.09
Cloud-Only	**5.13**	**296.27**	**8.14**

Exp. 1: Privacy–Efficiency Trade-off



(a) Energy Consumption (b) Completion Time (c) Inference Quality

Exp. 2: Robustness to Privacy Budget

Model	PRISM			Edge-Only			Cloud-Only		
	Ct.(s)	Ec.(J)	IQ.	Ct.(s)	Ec.(J)	IQ.	Ct.(s)	Ec.(J)	IQ.
(L1) GPT-4o + *(S1)* Phi-3.5-mini-3.5B	8.29	683.83	7.00	**15.98**	**1393.88**	5.19	5.22	**271.27**	**8.28**
(L1) GPT-4o + *(S2)* Qwen1.5-1.8B	**7.08**	**632.24**	6.91	17.29	1540.33	**5.59**	-	-	-
(L1) GPT-4o + *(S3)* Stablelm-2-zephyr-1.6B	7.34	657.88	7.16	18.57	1627.46	4.94	-	-	-
(L1) GPT-4o + *(S4)* Tinyllama-1.1B	7.35	653.62	5.28	19.50	1734.24	4.62	-	-	-
(L2) Qwen3-235B + *(S1)* Phi-3.5-mini-3.5B	8.59	738.88	**7.22**	-	-	-	**5.04**	321.28	8.01
(L2) Qwen3-235B + *(S2)* Qwen1.5-1.8B	8.60	739.59	7.06	-	-	-	-	-	-
(L2) Qwen3-235B + *(S3)* Stablelm-2-zephyr-1.6B	8.00	693.13	7.19	-	-	-	-	-	-
(L2) Qwen3-235B + *(S4)* Tinyllama-1.1B	8.11	698.10	7.19	-	-	-	-	-	-

Exp. 3: Adaptability Across Cloud–Edge Model Pairs

CEED

Dilemma: Limited resource at the Edge VS. real-time high-accuracy inference

Edge devices (IoT, sensors, mobile) are resource-limited (compute, memory, bandwidth) but need real-time DNN inference.

Deeper DNN models can achieve higher accuracy, but inference latency also grows a lot.

Existing approach:

- Model quantization and pruning :reduce model size, but result in reduced accuracy. (Model)
- Early-Exit Neural Networks (EENN): add intermediate exits → reduce latency by exiting early. (Infra)

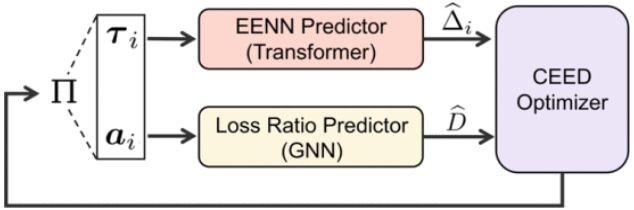
Key Challenges:

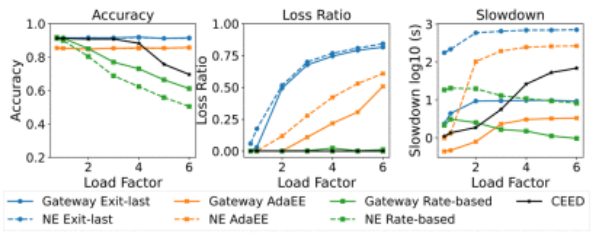
- **Threshold selection:** early-exit confidence thresholds impact latency, accuracy, and downstream load—hard to choose.
- **Chain assignment:** multiple edge devices & early-exit uncertainty → combinatorial system paths.
- **Legacy methods:** heuristic might be suboptimal or RL **don't scale** with multi-gateway chains + multi-layer thresholds.

Penn Engineering

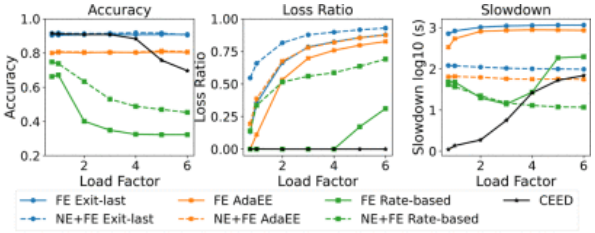
[1] Chen Yichong, Zifeng Niu, Manuel Roveri, and Giuliano Casale. "CEED: Collaborative Early Exit Neural Network Inference at the Edge." In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*, pp. 1-10. IEEE, 2025.

CEED





(a) Gateway-based and NE-based methods



(b) FE-based and NE+FE-based methods

CEED Framework

- **Two Predictors**
  - *EENN Predictor (Transformer)*: Predicts early-exit probability & inference accuracy from confidence thresholds
  - *Loss Predictor (GNN)*: Fast predicts system data loss rate from chain assignment policies
- **Joint Optimizer**
  - *optimizes: Confidence thresholds (τ) + Chain assignment (a)*
  - *Goal: Maximize accuracy while capping loss rate*

Key Results

- **Better Balance**: ~0.91 high accuracy (even under heavy load) + lower data loss
- **Lower Latency**: Faster inference than "no early-exit" baselines (stable on large models like ResNet101)
- **High Scalability**: Works for multi-gateway/multi-chain setups (solves RL's scalability issue)

CEED

Strengths

- *Novel Joint Optimzation:* Integrates accuracy and system loss into a differentiable reward function.
- *Strong Scalability:* Adopts a modular predictor–optimizer design, enabling seamless adaptation to new edge systems or additional devices by retraining the Transformer and GNN.

Weakness

- *Fixed-k Sparse Routing:* Random initialization of routing scores $s_{i,j}$, combined with fixed-k sparse softmax might skip the good chains in the beginning.
- *Static Training Assumption:* Changes in job distributions can reduce the accuracy of learned predictors, affecting overall system optimization.

CEED

Improvement Ideas

Two-stage Routing

1. *Dense Softmax (Exploration): Adopt full dense Softmax to iterate all candidate chains, eliminating initialization bias and identifying high-performance chains.*
2. *Sparse Top-k (Execution): Select top-k optimal chains for sparse Softmax, balancing inference efficiency and computational overhead.*

Test Time Adapt

1. *Drift Detection: Monitor 2 key deviation losses (Accuracy loss& Loss ratio loss)*
2. *Incremental Updates: Update the Transformer to handle accuracy drift and the GNN to capture changes in system loss ratios, using incremental learning with layer freezing.*

Thank you!

Penn Engineering