Ahmad, Jawad, et al., IEEE Open Journal of the Communications Society (2025)

A Stochastic Prototypical Network for Few-Shot Intrusion Detection in CAN-Based IoV Network

Department of Software, Sookmyung Women's University
Jisoo Kim

Table of Contents

01 Introduction

- Problem statement
- Challenges
- Proposed approach
- Evaluation results

ABSTRACT The Controller Area Network (CAN) acts as the backbone of intra-vehicle communication in modern Internet of Vehicles (IoV) systems, enabling real-time coordination among critical automotive subsystems. Despite its widespread adoption, CAN lacks essential security mechanisms such as encryption and message authentication, rendering it highly vulnerable to cyberattacks that can jeopardize vehicle safety and operational integrity. Developing an effective Few-Shot Learning (FSL)-based Intrusion Detection System (IDS) for CAN networks presents challenges due to data scarcity, noisy traffic, dynamic attack patterns, and the need for real-time efficiency. Existing FSL approaches often rely on deterministic models that struggle to capture the uncertainty and variability inherent in CAN network traffic. To address these challenges, we propose a Stochastic Prototypical Network based on a Random Neural Network (RaNN) for few-shot intrusion detection in CAN-based networks. RaNNs are inherently stochastic, enabling them to model uncertainty and variability in network traffic. By integrating RaNN with the prototypical network, the proposed framework computes stochastic prototypes that represent the distribution of normal and attack behaviors, improving robustness in noisy and dynamic environments. Additionally, the framework quantifies uncertainty in its predictions, enabling the system to flag ambiguous cases for further analysis, thereby reducing the risk of both false positives and negatives. The proposed approach demonstrates high classification performance across all FSL scenarios, achieving a maximum accuracy of 99.17% in a 15-shot configuration. The framework shows impressive computational efficiency with millisecond inference times and minimal training overhead, making it suitable for real-time deployment.

INDEX TERMS Internet of Vehicles, few-shot learning, controller area network (CAN), intrusion detection, stochastic prototypical network, RaNN.

01 Introduction

- Traditional IDS for CAN networks
  - → rule-based methods or machine learning models trained on large datasets
  - → struggle to detect novel or evolving attacks (especially when labeled attack data is scarce)
- Promising solution: **FSL!!** (Few-Shot Learning)
  - → can generalize effectively with only a handful of labeled examples
  - → However, existing FSL methods (ex. prototypical networks) often rely on **deterministic models** that may not adequately capture the uncertainty and variability inherent in CAN network traffic…

Let’s develop an “effective” FSL-based IDS for CAN networks!

01 Introduction

<Several challenges>

1. Data scarcity : obtaining labeled attack data is difficult and time-consuming
2. CAN network traffic is inherently noisy & highly variable
3. The dynamic nature of cyber threats necessitates that IDS adapt quickly to new attack types, which is challenging with limited labeled data
4. CAN networks operate in real time, necessitating lightweight and efficient detection mechanisms

→ Existing FSL approaches often fail to address these, as the typically rely on **deterministic models** that do not account for the uncertainty & variability in network traffic (since they rely on fixed points!!)

Random Neural Network(RaNN)-based Prototypical Network for few-shot intrusion detection in CAN-based in-vehicle networks

RaNN → inherently stochastic & capable of modeling uncertainty
→ well-suited for capturing the probablistic behavior of CAN network traffic
→ enables the model to handle noise & variability in the data better

01 Introduction

<Key Contributions>

1. **Integration of Stochastic Modeling(RaNN) with FSL**
   - Model can handle uncertainty, noise, and variability in the data more effectively, making it more robust and reliable for intrusion detection in dynamic environments.
2. **Uncertainty-aware Intrusion Detection** : incorporation of uncertainty quantification into the FSL process
   - Leveraging probablistic outputs of RaNN → provides confidence estimates for its intrusion predictions
   - Ability to quantify uncertainty → enables to flag ambiguous cases for further analysis, reducing the risk of both FP & FN
3. **Adaptability to Evolving Threats**
   - Can generalize effectively to previously unseen attacks by combining the rapid adaptation capabilities of FSL with the dynamic modeling strengths of RaNNs.

02 Related Work

1. Zhao et al.
   - Proposal : **Hybrid FSL-based scheme IDS**
   - Leverage 3D convolutional neural network for feature extraction, then establish a feature comparison network for learning and discrimination
   - Evaluation : Car-Hacking, ICSX2012 dataset → good attack detection accuracy for various few-shot configurations
2. Xu et al.
   - Proposal : **Conditional GAN-based FSL scheme**
   - Incorporated conditional GAN model with multiple generators & discriminators + adaptive sampling data technique → false positive rate reduced
   - Investigation : F2MD vehicle network simulation platform
3. Lu et al.
   - Proposal : **few-shot-based MAML approach** for intrusion detection in generic IoT applications
   - Leverage the key concept of meta-learning & Distribute the training process into two stages based on MAML
   - Investigation : Five open source IoT security datasets under different few-shot configurations
4. Yang et al.
   - Proposal : **A novel IDS framework**
   - **Flow data encoding + Feature fusion + Episode training**
   - Utilize Task generator to split the dataset into separate tasks and perform episode training → the extraction module and distance metric incorporated
5. Zhang et al.
   - Proposal : **A novel approach for generating network traffic samples**
   - Utilize the Conditional Denoising Diffusion Probabilistic Model → Developed integrated FSL-based intrusion detection technique by leveraging **CNN+BiGRU** based on synthetic data generation in the first stage
   - Evaluation : two open-source datasets

02 Related Work

- To summarize, in previous FSL IDS studies…
  - primarily focus on generic IoT/network datasets, but **studies on IoV(CAN) are rare.**
  - are based on **"deterministic"** models → cannot express the noise/variability/uncertainty of CAN traffic

The existing FSL-based studies primarily focus on generic IoT applications and have not been thoroughly tested with real-world vehicular network datasets. Furthermore, the existing FSL-based scheme primarily relies on deterministic models, which struggle to capture the uncertainty and variability inherent in vehicular network traffic. To overcome these limitations, we introduced a stochastic prototypical network few-shot intrusion detection in CAN-based in-vehicle networks.

03 Preliminaries

A. CONTROLLER AREA NETWORK

Let $N = \{n_1, n_2, \ldots, n_k\}$ represent the set of nodes in a CAN system. Each node n_i transmits messages m_i over the shared bus. A message m_i is formally defined as:

$$m_i = (ID_i, D_i, L_i), \tag{1}$$

where: $ID_i \in \mathbb{Z}^+$ is the unique identifier of the message, $D_i \in \{0, 1\}^d$ is the data payload of length d, and $L_i \in \mathbb{Z}^+$is the length of the data payload in bytes.

Mathematically, if two nodes n_i and n_j transmit messages m_i and m_j simultaneously, the message with the smaller ID is granted access to the bus:

$$\text{Arbitration}\left(m_i, m_j\right) = \begin{cases} m_i & \text{if } ID_i < ID_j \\ m_j & \text{otherwise} \end{cases} \tag{2}$$

The state $B(t)$ is determined by the superposition of all transmitted messages at time t:

$$B(t) = \bigoplus_{i=1}^{k} m_i(t) \tag{3}$$

where $\bigoplus$ denotes the logical OR operation, and $m_i(t)$ is the message transmitted by node n_i at time t.

- **Arbitration mechanism** (to resolve conflicts when multiple nodes attempt to transmit simultaneously)
  - based on message ID : lower ID values have higher priority

B. FEW-SHOT LEARNING WITH PROTOTYPICAL NETWORKS

Given a support set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where x_i is an input instance and y_i is its corresponding class label, Prototypical Networks compute a prototype c_k for each class k as the mean of the embedded support instances:

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\theta(x_i) \tag{4}$$

where S_k is the set of support instances belonging to class k, and f_θ is an embedding function parameterized by θ. For a query instance x, the network predicts its class by computing the distance between the embedded query and each class prototype:

$$p(y = k \mid x) = \frac{\exp\left(-d(f_\theta(x), c_k)\right)}{\sum_{k'} \exp\left(-d(f_\theta(x), c_{k'})\right)} \tag{5}$$

where $d(\cdot, \cdot)$ is a distance metric, typically Euclidean distance.

- **Prototypical network**
  - one of the popular FSL approaches that leverage metric learning to classify instances based on their proximity to class prototypes in a learned embedding space.

03 Preliminaries

1. Generate prototype **ck** for each class with Support set S

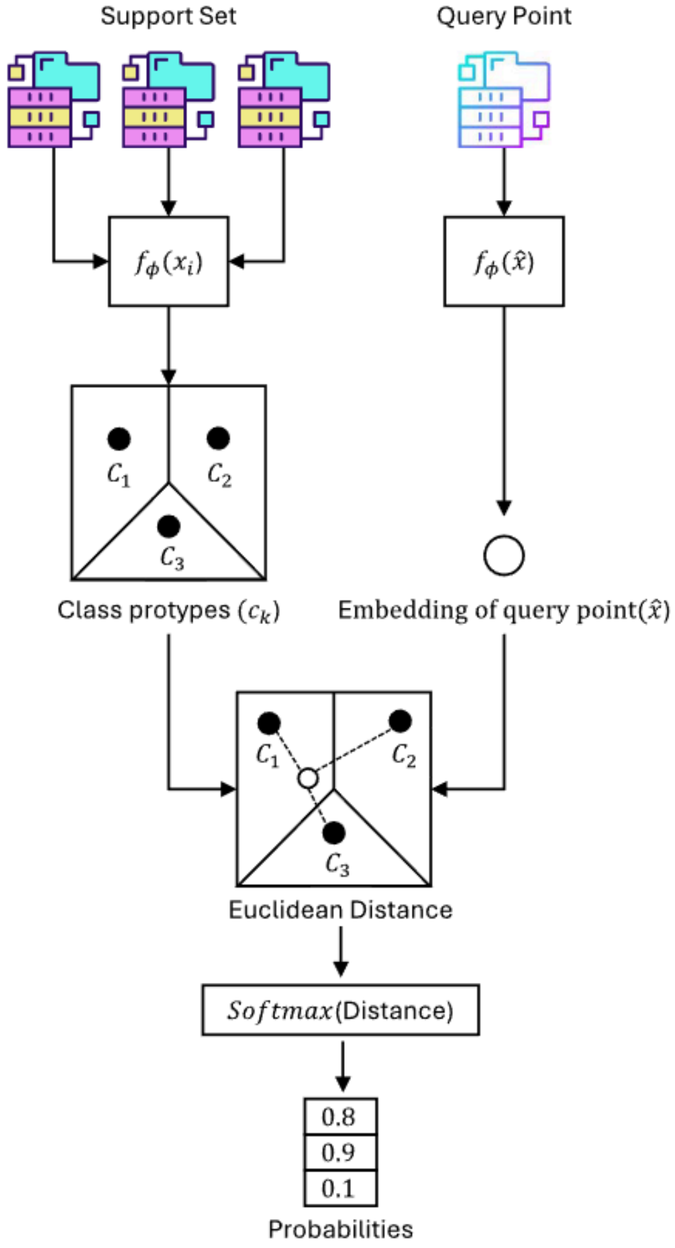


FIGURE 2. **Workflow of prototypical network.**

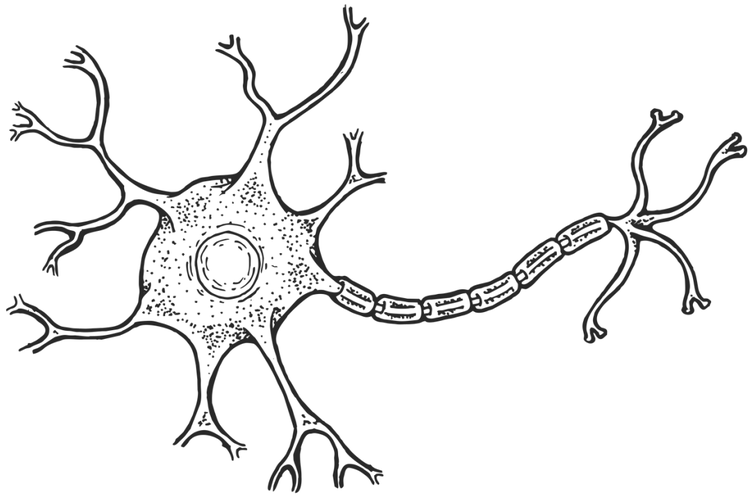
2. Classify the distance to each ck using softmax to probabilistically classify it as p **(y = k | x)**

3. (While training) Compute the loss by comparing it with the query answer and update the embedding f_θ.

03 Preliminaries

C. RANDOM NEURAL NETWORKS

- M : Number of neurons
- Excitation probability qi : The probability that neuron i is in the “exited” state (the basis of the output)
- $\lambda i+$: The arrival rate of the excitation signal
- $\lambda i-$: The arrival rate of the inhibition signal



The RaNN consists of M neurons, where each neuron i is characterized by its excitation probability q_i, defined as the probability that neuron i is in an excited state. The excitation probability q_i is governed by the following equation:

$$q_i = \frac{\lambda_i^+}{r_i + \lambda_i^-} \tag{6}$$

where: λ_i^+ is the arrival rate of excitatory signals to neuron i, λ_i^- is the arrival rate of inhibitory signals to neuron i, and r_i is the firing rate of neuron i.

The arrival rates λ_i^+ and λ_i^- are determined by the network’s connectivity and the activity of other neurons. Specifically:

$$\lambda_i^+ = \sum_{i=1}^{M} q_j w_{ji}^+ + \Lambda_i^+, \tag{7}$$

$$\lambda_i^- = \sum_{j=1}^{M} q_j w_{ji}^- + \Lambda_i^- \tag{8}$$

C. RANDOM NEURAL NETWORKS

where:

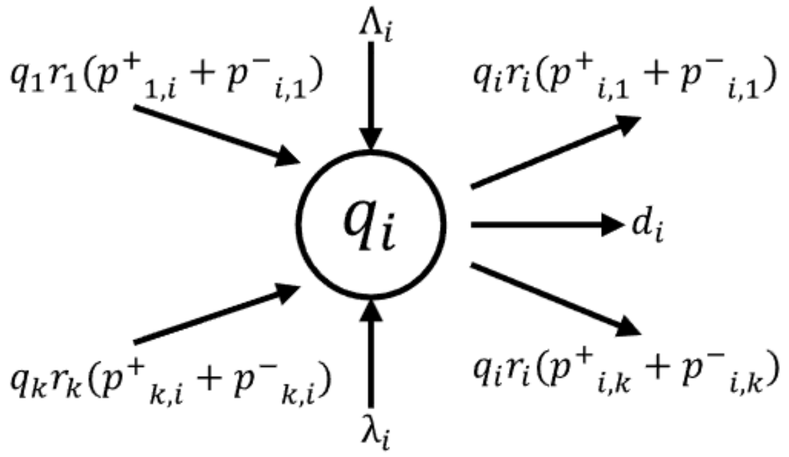
- w_{ji}^{+} and w_{ji}^{-} are the excitatory and inhibitory weights, respectively, from neuron j to neuron i,
- Λ_i^{+} and Λ_i^{-} are external excitatory and inhibitory input rates to neuron i.

The firing rate r_i is a function of the neuron's internal state and can be expressed as:

$$r_i = \frac{1}{\tau_i} \tag{9}$$

where τ_i is the mean inter-spike interval of neuron i.

- **Firing rate** ri: the average rate at which a neuron i emits spikes (= $1/\tau i$, τi: average spike interval)
  - The shorter the τi (=more frequent), the higher the ri



$w^+_{j,i} = r_i p^+_{i,j}$

$w^-_{j,i} = r_i p^-_{i,j}$ → Non-negative rates spike emission
Excitatory / Inhibitory

$$d_i + \sum_{j=1}^{n} \left[p^+_{i,j} + p^-_{i,j} \right] = 1$$

$\mathbf{\Lambda_i}$: Arrival rate of external excitatory signal
$\boldsymbol{\lambda_i}$: Arrival rate of external inhibitory signal
$\mathbf{q_i}$: Probability neuron excited
$\mathbf{r_i}$: Rate neuron fires spikes
$\mathbf{p^+}$: Probability excitatory signal neuron i –> neuron j
$\mathbf{p^-}$: Probability inhibitory signal neuron i –> neuron j
$\mathbf{d_i}$: Probability signal leaves network

FIGURE 3. **Basic architecture of RaNN [20].**

04 Design of the proposed architecture

A. PROBLEM FORMULATION AND DESIGN OBJECTIVES

- Input space X : The time-series feature space of CAN bus traffic
  - Each observation $x \in X$: time series of CAN messages
- Support set S : **N = K X C** (C: number of class, K : shot per each class)
  - Task : Classify a query sample xq into one of the classes {0, 1, ..., C}, where class 0 denotes normal traffic

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

<Primary challenges>

1. Data Scarcity
2. Noisy and Variable Traffic
3. Evolving Threat Landscape
4. Real-Time Efficiency

<Design Objectives>

1) Uncertainty Awareness: The model should estimate a confidence score $\sigma(x_q) \in [0, 1]$ for each prediction. Predictions with $\sigma(x_q) < \tau$ (a predefined threshold) should be flagged as ambiguous for further review.
2) Noise Robustness: For a perturbed input $x + \delta$ where $\|\delta\|_\infty \leq \epsilon$, the prediction should remain stable:

$$\|f(x + \delta) - f(x)\|_2 \leq \gamma, \quad \forall x \in X$$

where γ is a small constant.

3) Few-Shot Adaptability: Upon encountering a new attack class Y_{new} with K examples, the system should incorporate it in $O(K)$ time without retraining the entire model.

04 Design of the proposed architecture

B. THREAT MODEL AND ASSUMPTIONS

<Adversary A's Goal>

- Inject or replay malicious messages $x' \in X$, such that the observed distribution p(x′ | y′) deviates from normal behavior p(x | y = 0) without detection

<Capabilities Assumption of Adversary A>

- **Physical Access (A_phys)** : Can perform message injection, via diagnostic ports or compromised ECUs
- **Message Injection (A_inj)** : Can transmit forged messages m′ = (ID′, D′, L′) onto the bus, attempting to spoof the behavior of legitimate nodes
- **DoS/Fuzzing (A_dos)** : May flood the bus with high-priority or malformed messages to disrupt normal operations

<System Assumption>

- **Passive observer IDS** : Doesn't interfere with or alter CAN communication, receiving a stream of messages xt $\in X$ and performs real-time classification $f(xt) \in \{0,\ldots,C\}$
- No possessing ECU firmware-level access
- Detection with **Stochastic Prototypical Network**

04 Design of the proposed architecture - Overall

C. MATHEMATICAL MODELING OF THE PROPOSED ARCHITECTURE

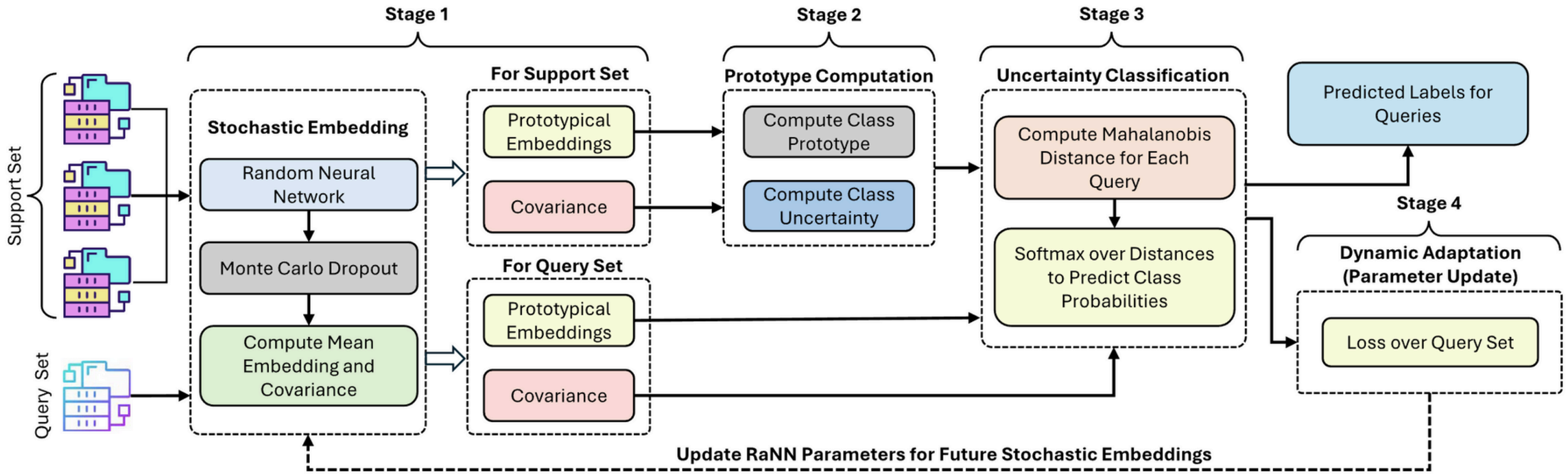


FIGURE 4. **Block diagram of the proposed architecture.**

04 Design of the proposed architecture - Overall

C. MATHEMATICAL MODELING OF THE PROPOSED ARCHITECTURE

<4 integrated stages>

1) **Stochastic Embedding** (RaNN-based feature extraction)
2) **Prototype Computation** (Class distribution modeling)
3) **Uncertainty-Aware Classification** (Mahalanobis distance with MC dropout)
4) **Dynamic Adaptation** (Few-shot prototype updates)

Algorithm 5 Few-Shot Training for RaNN-Based Prototypical Network

Input:
  Training episodes $\{\mathcal{E}_i\}$, where $\mathcal{E}_i = (\mathcal{S}_i, \mathcal{Q}_i)$
  RaNN parameters θ, learning rate η
Output: Trained model f_θ

Meta-Training Phase:
for each episode $\mathcal{E}_i$ **do**
  1. Stochastic Embedding:
  for each $x_j \in \mathcal{S}_i \cup \mathcal{Q}_i$ **do**
    $z_j \sim q_\theta(z \mid x_j) = \text{RaNN}(x_j; \theta)$
    $\bar{z}_j = \frac{1}{T} \sum_{t=1}^{T} z_j^{(t)}$
    $\Sigma_j = \text{Cov}(\{z_j^{(t)}\})$
  end for
  2. Prototype Computation:
  for each class $c \in \mathcal{E}_i$ **do**
    $\mu_c = \frac{1}{|\mathcal{S}_i^c|} \sum_{j \in \mathcal{S}_i^c} \bar{z}_j$
    $\Sigma_c = \frac{1}{|\mathcal{S}_i^c|} \sum_{j \in \mathcal{S}_i^c} \left(\Sigma_j + (\bar{z}_j - \mu_c)(\bar{z}_j - \mu_c)^T\right)$
  end for
  3. Uncertainty-Aware Classification:
  for each query $(x_q, y_q) \in \mathcal{Q}_i$ **do**
    $d_c^2 = (\bar{z}_q - \mu_c)^T (\Sigma_c + \Sigma_q + \epsilon I)^{-1} (\bar{z}_q - \mu_c)$
    $p(y_q = c) = \frac{\exp(-d_c^2)}{\sum_{c'} \exp(-d_{c'}^2)}$
  end for
  4. Parameter Update:
  $\theta \leftarrow \theta - \eta \nabla_\theta \sum_{(x_q, y_q) \in \mathcal{Q}_i} \mathcal{L}(p(y_q), y_q)$
end for

04 Design of the proposed architecture - 1

- Input
  - All the support and query samples for this episode, $x \in S \cup Q$
- What
  - Pass each sample through the RaNN dynamics to obtain **"probabilistic embedding samples"**, from which we derive the **mean** $\bar{z}$ and **covariance Σ** (=uncertainty)
- Output
  - $\bar{z}j$, Σj for each sample
- Summary
  - This step turns each sample into a **"small distribution (mean + spread)"** rather than a "point"

Algorithm 1 RaNN-Based Stochastic Embedding (Stage 1)

Require: CAN traffic sample $x = \{x_1, \ldots, x_T\}$, RaNN parameters $\theta = (W, b, \lambda^+, \lambda^-, w^+, w^-)$

Ensure: Stochastic embedding $z \sim q_\theta(z \mid x)$ and uncertainty $\mathrm{Var}(z \mid x)$

Initialize:

Neuron excitation probabilities $\mathbf{q} = [q_1, \ldots, q_n]^T \leftarrow \mathbf{0}$

External input rates $\Lambda_i \leftarrow \phi(x)_i$ for $i = 1, \ldots, n$

Compute neuron excitations:

repeat

  for $i = 1$ to n **do**

$$q_i \leftarrow \frac{\Lambda_i + \sum_{j=1}^{n} q_j w_{ji}^+}{r_i + \lambda_i^- + \sum_{j=1}^{n} q_j w_{ji}^-}$$

  end for

until $\|\mathbf{q}^{(k+1)} - \mathbf{q}^{(k)}\|_2 < \epsilon$

Generate stochastic embedding:

Sample $\epsilon \sim \mathcal{N}(0, \mathrm{diag}(\Sigma))$

$z \leftarrow \sigma(W\mathbf{q} + b + \epsilon)$

Compute uncertainty:

$\mathrm{Var}(z \mid x) \leftarrow \mathrm{diag}(\Sigma) + J_q \Sigma_q J_q^T$

return z, $\mathrm{Var}(z \mid x)$

04 Design of the proposed architecture - 2

- Input
  - Samples ($z^{-}i$,Σi) and labels obtained from the support set
- What
  - Group classes together, compute the class mean μc (average of means) and class covariance Σc (sample uncertainty + intra-class variance), and add numerical stabilization to create $\Sigma\sim c$.
- Output
  - The probability prototype for class c, $Pc=N(\mu c,\Sigma\sim c)$.
- Summary
  - Create a large **"class distribution"** with support set (by combining the smaller distributions from stage1)

Algorithm 2 Stochastic Prototype Computation (Stage 2)

Require: Support set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N}$ with labels $y_i \in \{0, 1, \ldots, C\}$

RaNN embedding function $f_\theta: \mathcal{X} \to \mathbb{R}^d$

Regularization parameter $\epsilon > 0$

Ensure: Class prototypes $\{\mathcal{P}_c = \mathcal{N}(\mu_c, \tilde{\Sigma}_c)\}_{c=1}^{C}$

Initialize:

for $c = 1$ to C **do**

  $\mathcal{S}_c \leftarrow \{(x_i, y_i) \in \mathcal{S} \mid y_i = c\}$

end for

Compute class prototypes:

for $c = 1$ to C **do**

  Compute mean prototype:

  $\mu_c \leftarrow \frac{1}{|\mathcal{S}_c|} \sum_{(x_i,y_i)\in\mathcal{S}_c} \mathbb{E}[f_\theta(x_i)]$

  Compute covariance prototype:

  $\Sigma_c \leftarrow \frac{1}{|\mathcal{S}_c|} \sum_{(x_i,y_i)\in\mathcal{S}_c} \Big(\mathrm{Cov}(f_\theta(x_i)) + (\mathbb{E}[f_\theta(x_i)] - \mu_c)(\mathbb{E}[f_\theta(x_i)] - \mu_c)^T\Big)$

  Regularize covariance:

  $\tilde{\Sigma}_c \leftarrow \Sigma_c + \epsilon I_d$

  Store prototype:

  $\mathcal{P}_c \leftarrow \mathcal{N}(\mu_c, \tilde{\Sigma}_c)$

end for

Return prototypes: return $\{\mathcal{P}_1, \ldots, \mathcal{P}_C\}$

04 Design of the proposed architecture - 3

- Input
  - Query sample xq and the class prototypes Pc we just created in stage 2
- for each class, compute :

$$d^2(z_q, \mathcal{P}_c) = (\bar{z}_q - \mu_c)^T \left(\tilde{\Sigma}_c + \Sigma_q + \epsilon I\right)^{-1} (\bar{z}_q - \mu_c) + \ln\left|\tilde{\Sigma}_c + \Sigma_q\right| \quad (21)$$

$$\text{Confidence}_c = \exp\left(-\frac{d^2(z_q, \mathcal{P}_c)}{\tau}\right), \quad \tau > 0 \quad (22)$$

- Output
  - The confidence and predicted label for each class
- Summary
  - **Generate a loss and update** $\theta \rightarrow$ the embedding model $f\theta$ gradually improves to correctly match the query
  - Compute robust distance measure between query embeddings and class prototypes

Algorithm 3 Uncertainty-Aware Distance Metric (Stage 3)

Require: Query sample x_q with unknown label
Class prototypes $\{\mathcal{P}_c = \mathcal{N}(\mu_c, \tilde{\Sigma}_c)\}_{c=1}^{C}$ from Stage 2
RaNN embedding function f_θ with MC dropout (T samples)
Temperature parameter $\tau > 0$
Ensure: Confidence scores $\{\text{Confidence}_c\}_{c=1}^{C}$ and predicted class
Compute query embedding distribution:
for $t = 1$ to T **do** MC Dropout T times
  $z_q^{(t)} \leftarrow f_{\theta^{(t)}}(x_q)$ to collect embedding samples
end for
$\bar{z}_q \leftarrow \frac{1}{T}\sum_{t=1}^{T} z_q^{(t)}$
$\Sigma_q \leftarrow \frac{1}{T}\sum_{t=1}^{T}(z_q^{(t)} - \bar{z}_q)(z_q^{(t)} - \bar{z}_q)^T$
Compute uncertainty-aware distances:
for $c = 1$ to C **do**
  $\Sigma_{\text{total}} \leftarrow \tilde{\Sigma}_c + \Sigma_q + \epsilon I$
  $d_c^2 \leftarrow (\bar{z}_q - \mu_c)^T \Sigma_{\text{total}}^{-1} (\bar{z}_q - \mu_c) + \ln|\Sigma_{\text{total}}|$
  $\text{Confidence}_c \leftarrow \exp\left(-\frac{d_c^2}{\tau}\right)$
end for
Predict class:
$c^* \leftarrow \arg\max_c \text{Confidence}_c$
Return results: return $\{\text{Confidence}_c\}_{c=1}^{C}, c^*$

04 Design of the proposed architecture - 4

- Input
  - K-shot samples of the new class & existing prototypes & and a (fixed) embedding $f\theta$
- What
  - Create μnew and Σnew from K instances
  - To avoid overconfidence due to limited data, weight the average covariance Σglobal of the existing classes to obtain Σ~new.
  - Then, store the new prototype Pnew in memory (smoothed by β)
- Output
  - The updated prototype set {$P1$,...,PC, Pnew}
- Summary
  - Once got a few samples, create a new distribution, put it in, and run it right away.

Algorithm 4 Few-Shot Adaptation for New Attacks

Require: New attack samples $\mathcal{S}_{\text{new}} = \{(x_i, c_{\text{new}})\}_{i=1}^{K}$ (K-shot)

Existing prototypes $\{\mathcal{P}_c = \mathcal{N}(\mu_c, \tilde{\Sigma}_c)\}_{c=1}^{C}$

RaNN embedding function f_θ (frozen)

Global covariance $\Sigma_{\text{global}} = \frac{1}{C}\sum_{c=1}^{C}\tilde{\Sigma}_c$

Mixing coefficient $\alpha \in [0, 1]$, memory factor $\beta \in [0, 1)$

Ensure: Updated prototypes $\{\mathcal{P}_c\}_{c=1}^{C+1}$ including the new class

Compute new class statistics:

for each $(x_i, c_{\text{new}}) \in \mathcal{S}_{\text{new}}$ **do**

$\bar{z}_i \leftarrow \mathbb{E}[f_\theta(x_i)]$

$\Sigma_i \leftarrow \text{Cov}(f_\theta(x_i))$

end for

Initialize new prototype:

$\mu_{\text{new}} \leftarrow \frac{1}{K}\sum_{i=1}^{K}\bar{z}_i$

$\Sigma_{\text{new}} \leftarrow \frac{1}{K}\sum_{i=1}^{K}\left[\Sigma_i + (\bar{z}_i - \mu_{\text{new}})(\bar{z}_i - \mu_{\text{new}})^T\right]$

Regularize covariance: average existing covariance

$\tilde{\Sigma}_{\text{new}} \leftarrow \alpha\Sigma_{\text{new}} + (1-\alpha)\Sigma_{\text{global}} + \epsilon I$

Update prototype memory:

$\mathcal{P}_{\text{new}} \leftarrow \mathcal{N}(\mu_{\text{new}}, \tilde{\Sigma}_{\text{new}})$

Update prototype memory with exponential decay

$\mathcal{M} \leftarrow \beta\mathcal{M} + (1-\beta)\mathcal{P}_{\text{new}}$

Return updated prototypes: return $\{\mathcal{P}_1, \ldots, \mathcal{P}_C, \mathcal{P}_{\text{new}}\}$

04 Design of the proposed architecture - Overall(re)

- A meta-learning loop that repeats Stages 1 → 2 → 3 for each episode, updating θ with the query loss each time

- Input: Episode Ei = (Si, Qi)
- Stage 1: Passing samples of Si∪Qi through RaNN → bar_z, Σ
- Stage 2: Computing prototypes Pc using Si
- Stage 3: Computing confidence/prediction using Qi → Generating loss → Updating θ
- Iteration: Over multiple episodes, fθ learns a "few-shot embedding" that works well

- (Operating) Stage 4: Incorporating new classes using only K-shots

Algorithm 5 Few-Shot Training for RaNN-Based Prototypical Network

Input:
  Training episodes $\{\mathcal{E}_i\}$, where $\mathcal{E}_i = (\mathcal{S}_i, \mathcal{Q}_i)$
  RaNN parameters θ, learning rate η
Output: Trained model f_θ

Meta-Training Phase:
for each episode $\mathcal{E}_i$ **do**
  1. Stochastic Embedding:
  for each $x_j \in \mathcal{S}_i \cup \mathcal{Q}_i$ **do**
    $z_j \sim q_\theta(z \mid x_j) = \text{RaNN}(x_j; \theta)$
    $\bar{z}_j = \frac{1}{T}\sum_{t=1}^{T} z_j^{(t)}$
    $\Sigma_j = \text{Cov}(\{z_j^{(t)}\})$
  end for
  2. Prototype Computation:
  for each class $c \in \mathcal{E}_i$ **do**
    $\mu_c = \frac{1}{|\mathcal{S}_i^c|}\sum_{j \in \mathcal{S}_i^c} \bar{z}_j$
    $\Sigma_c = \frac{1}{|\mathcal{S}_i^c|}\sum_{j \in \mathcal{S}_i^c} \left(\Sigma_j + (\bar{z}_j - \mu_c)(\bar{z}_j - \mu_c)^T\right)$
  end for
  3. Uncertainty-Aware Classification:
  for each query $(x_q, y_q) \in \mathcal{Q}_i$ **do**
    $d_c^2 = (\bar{z}_q - \mu_c)^T(\Sigma_c + \Sigma_q + \epsilon I)^{-1}(\bar{z}_q - \mu_c)$
    $p(y_q = c) = \frac{\exp(-d_c^2)}{\sum_{c'} \exp(-d_{c'}^2)}$
  end for
  4. Parameter Update:
  $\theta \leftarrow \theta - \eta \nabla_\theta \sum_{(x_q, y_q) \in \mathcal{Q}_i} \mathcal{L}(p(y_q), y_q)$
end for

05 Experiments and Results

<Setup>

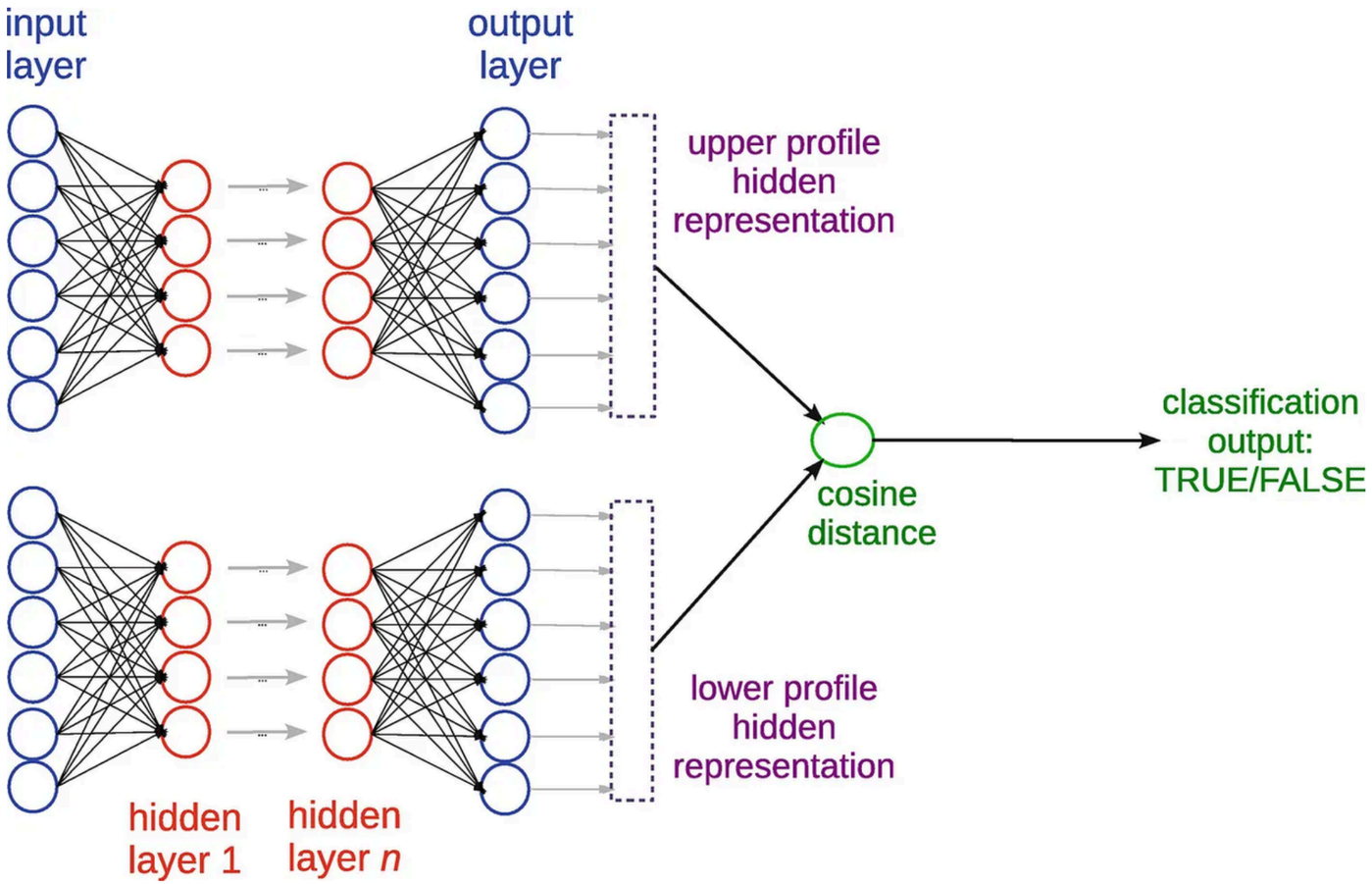
- **Dataset**
  - X-CANIDS → CAN signals collected under real-world driving and stationary conditions are decoded using DBC, resulting in 107 dynamic signal time series (200 Hz).
  - Five attack types (Fuzzing, Fabrication, Suspension, Masquerade, and Replay) are synthetically injected into the normal data for evaluation
  - Training is performed on data without attacks, and evaluation is performed on a mixed set

- **Few-shot episodes**
  - 5-way classification uses 5-shot, 10-shot, and 15-shot configurations
  - Episode-based: Prototypes are created using support, and performance is aggregated using queries

- **Baseline**
  - Representative few-shot methods in the metric learning family, such as Siamese and Relational Networks

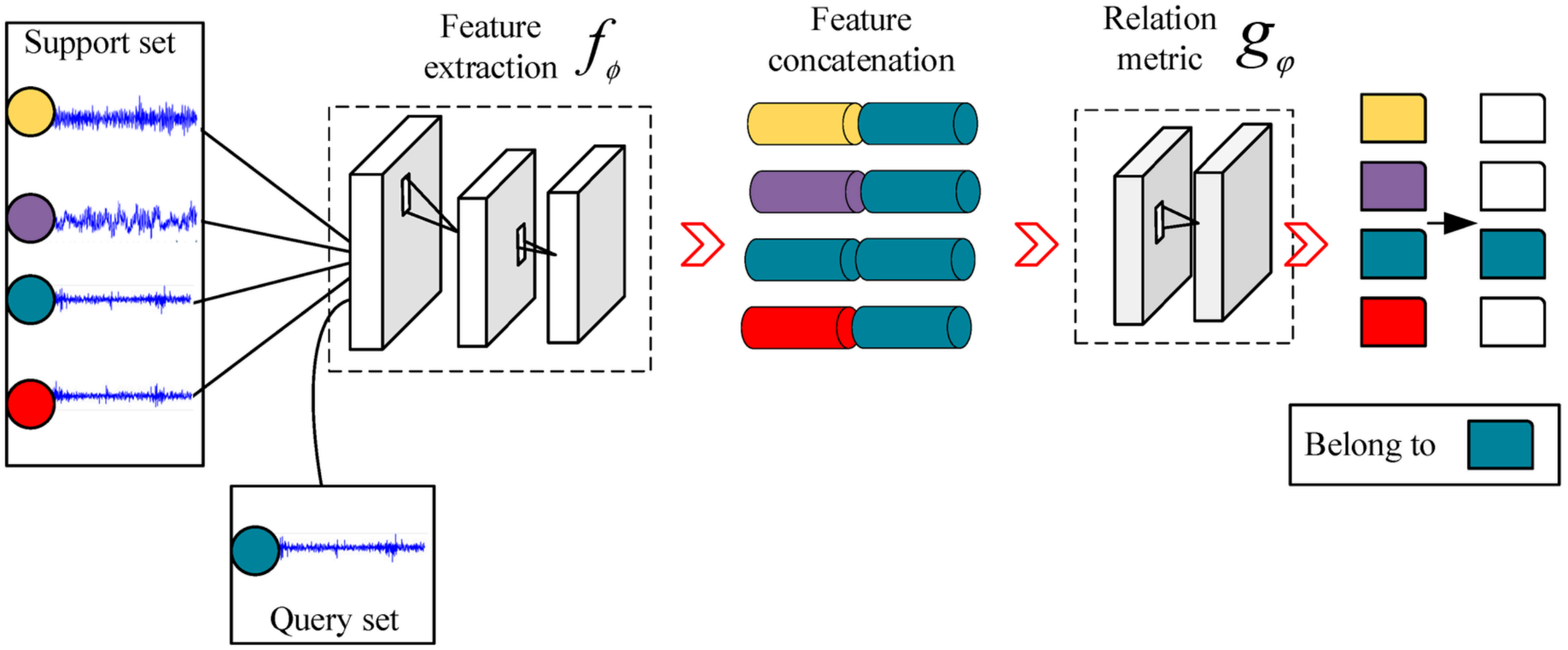
05 Experiments and Results

<Siamese network>



05 Experiments and Results

<Relation network>



05 Experiments and Results

TABLE 1. The system parameters for the proposed architecture.

Category	Parameter	Value/Setting	Purpose
Data Preprocessing	Test Size	0.2	Train-test split ratio
	Random State	42	Reproducibility seed
	Missing Values Handling	Median Imputation	Fill NA values
	Categorical Features	One-Hot Encoding	Convert categorical variables
	Feature Scaling	StandardScaler	Normalize features
Model Architecture	Hidden Layers	4	Units: [512, 256, 128, 64]
	Dropout Rate	0.2	MC-Dropout for uncertainty estimation
	Output Layer Activation	Softmax	Multi-class classification
	Noise StdDev	0.05	Gaussian noise in hidden layers
Training	Optimizer	Adam (lr=0.001)	Gradient-based optimization
	Loss Function	Categorical Crossentropy	Multi-class loss
	Batch Size	128	Samples per gradient update
	Epochs	20	Training iterations
	Validation Split	0.2	Internal validation during training
Uncertainty	Monte Carlo Samples	50	Stochastic forward passes
	Uncertainty Threshold	0.1	Flag ambiguous predictions
Few-Shot Learning	Support Shots per Class (k_shot)	5, 10, 15	Prototype examples per class
	Query Samples per Class (n_query)	20	Evaluation samples per class
	Classes per Episode (n_classes)	6	Classes sampled per few-shot episode
Evaluation	Metrics	Accuracy, Precision, Recall, F1	Weighted averages
	Confidence Interval	95%	Statistical significance
Visualization	Plots Generated	4 high-res PNGs (1200 DPI)	Confusion matrix, t-SNE, uncertainty distributions
System Monitoring	Memory/GPU Tracking	psutil, GPUtil	Resource usage before/after training
	Model Size Calculation	H5 file serialization	Disk-space measurement

05 Experiments and Results

B. CLASSIFICATION PERFORMANCE

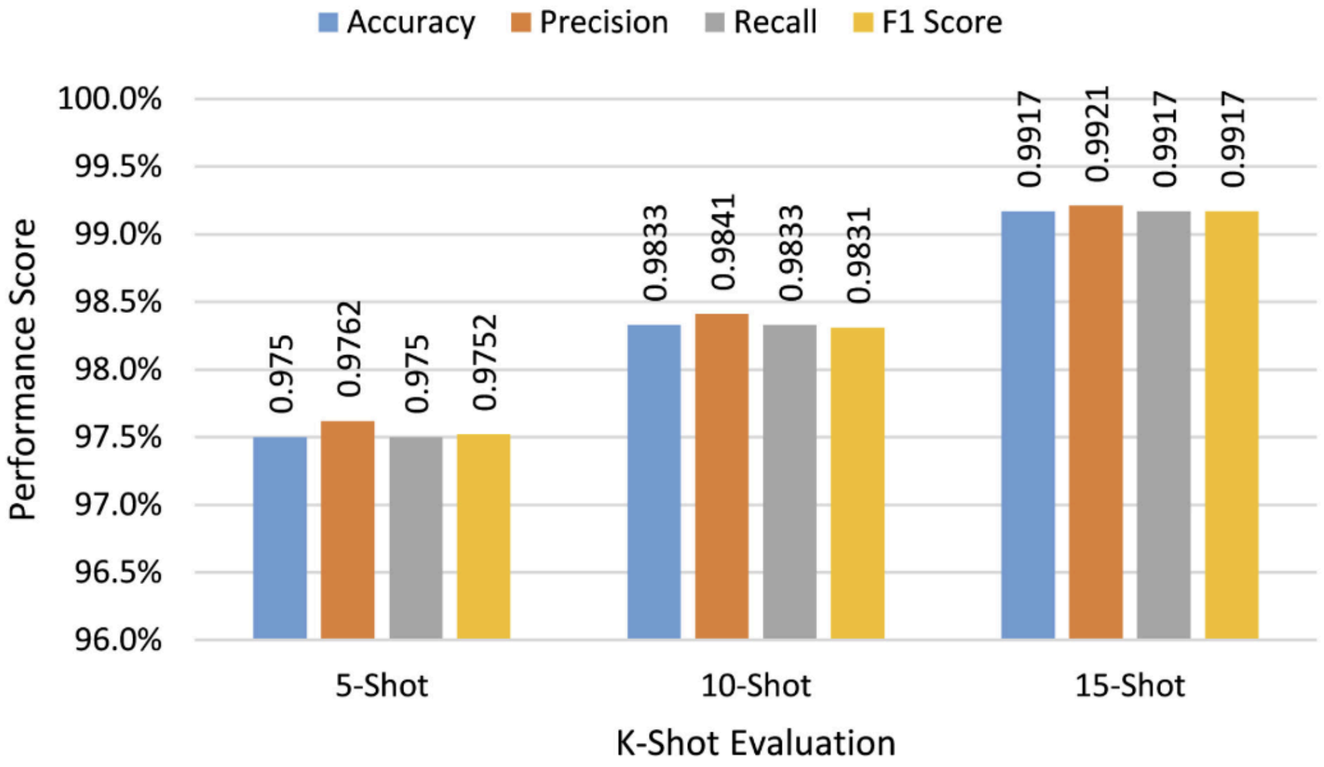
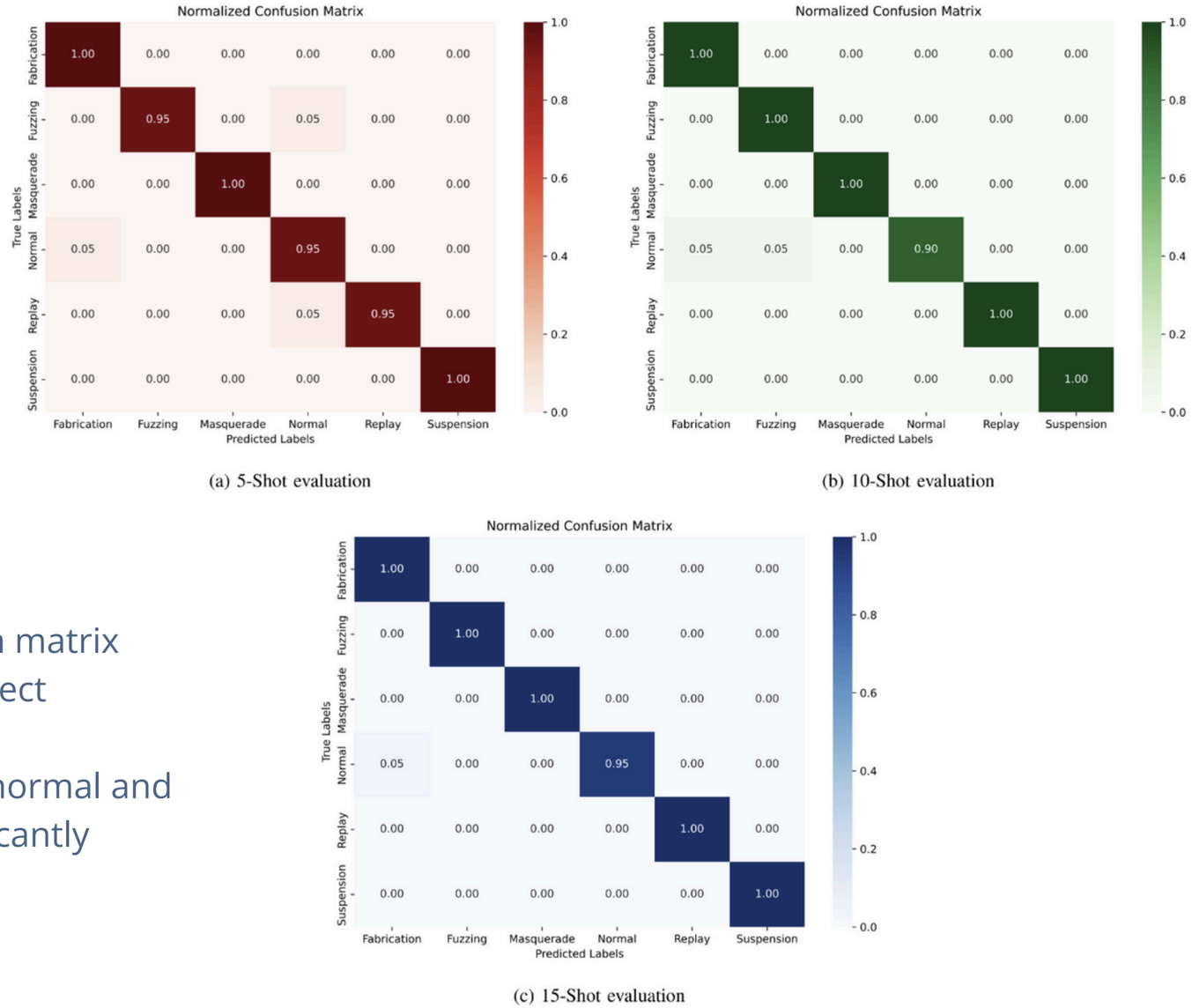


FIGURE 5. **Classification performance of the proposed framework.**

05 Experiments and Results



(a) 5-Shot evaluation

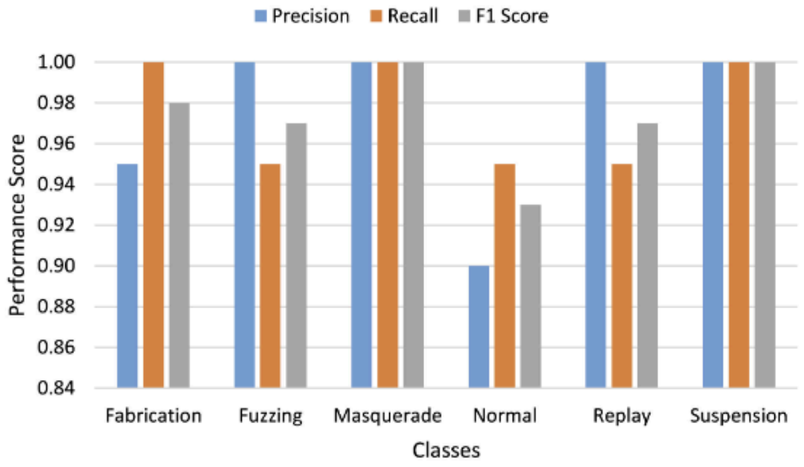
(b) 10-Shot evaluation

(c) 15-Shot evaluation

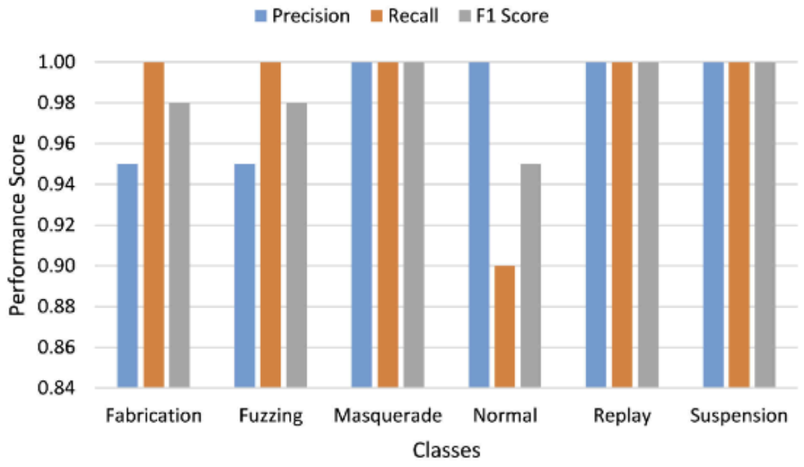
- As the number of shots increases, the confusion matrix fills only the diagonal lines and approaches perfect classification
- In particular, the ability to distinguish between normal and disguised attacks (masquerade, replay) is significantly improved.

FIGURE 6. **Confusion metrics for different k-shot evaluations.**

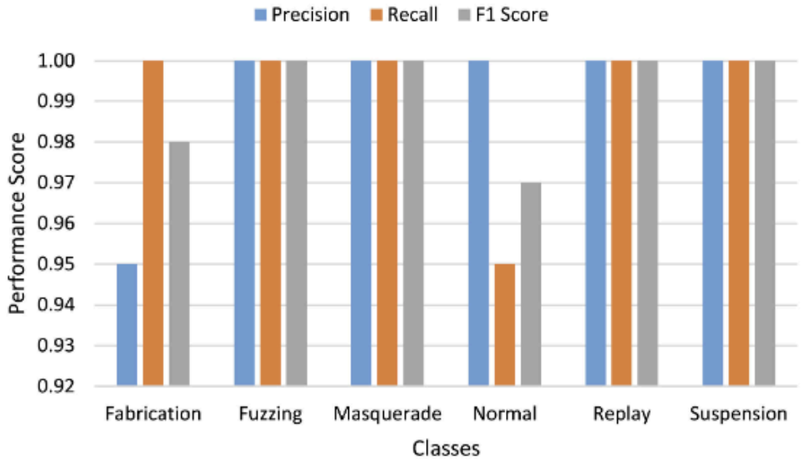
05 Experiments and Results



(a) 5-Shot evaluation



(b) 10-Shot evaluation



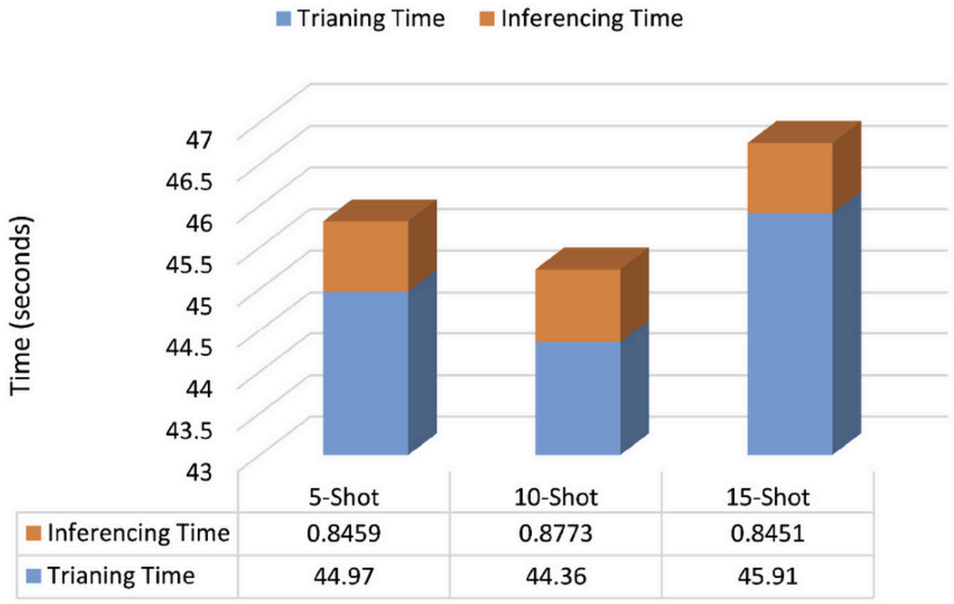
(c) 15-Shot evaluation

FIGURE 7. **Classification reports for different k-shot evaluations.**

05 Experiments and Results

C. COMPUTATIONAL EFFICIENCY

- The architecture is especially well-suited for scenarios that require frequent updates with new attack signatures while maintaining real-time detection performance, striking an optimal balance between computational efficiency and operational reliability



	5-Shot	10-Shot	15-Shot
Inferencing Time	0.8459	0.8773	0.8451
Trianing Time	44.97	44.36	45.91

FIGURE 8. **Computational cost analysis of the proposed framework.**

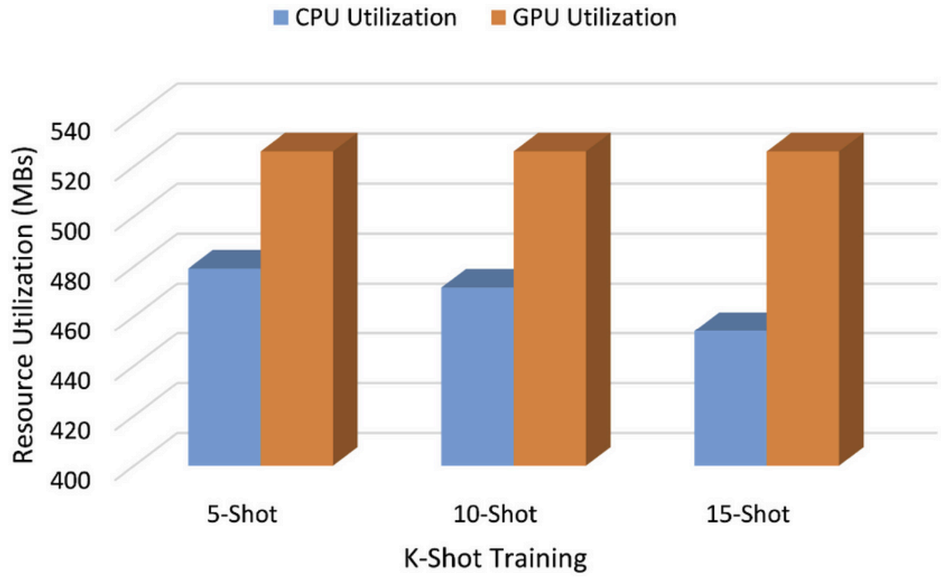


FIGURE 9. **Resource utilization of the proposed framework.**

- This stability reflects an optimal balance between model complexity and learning capacity

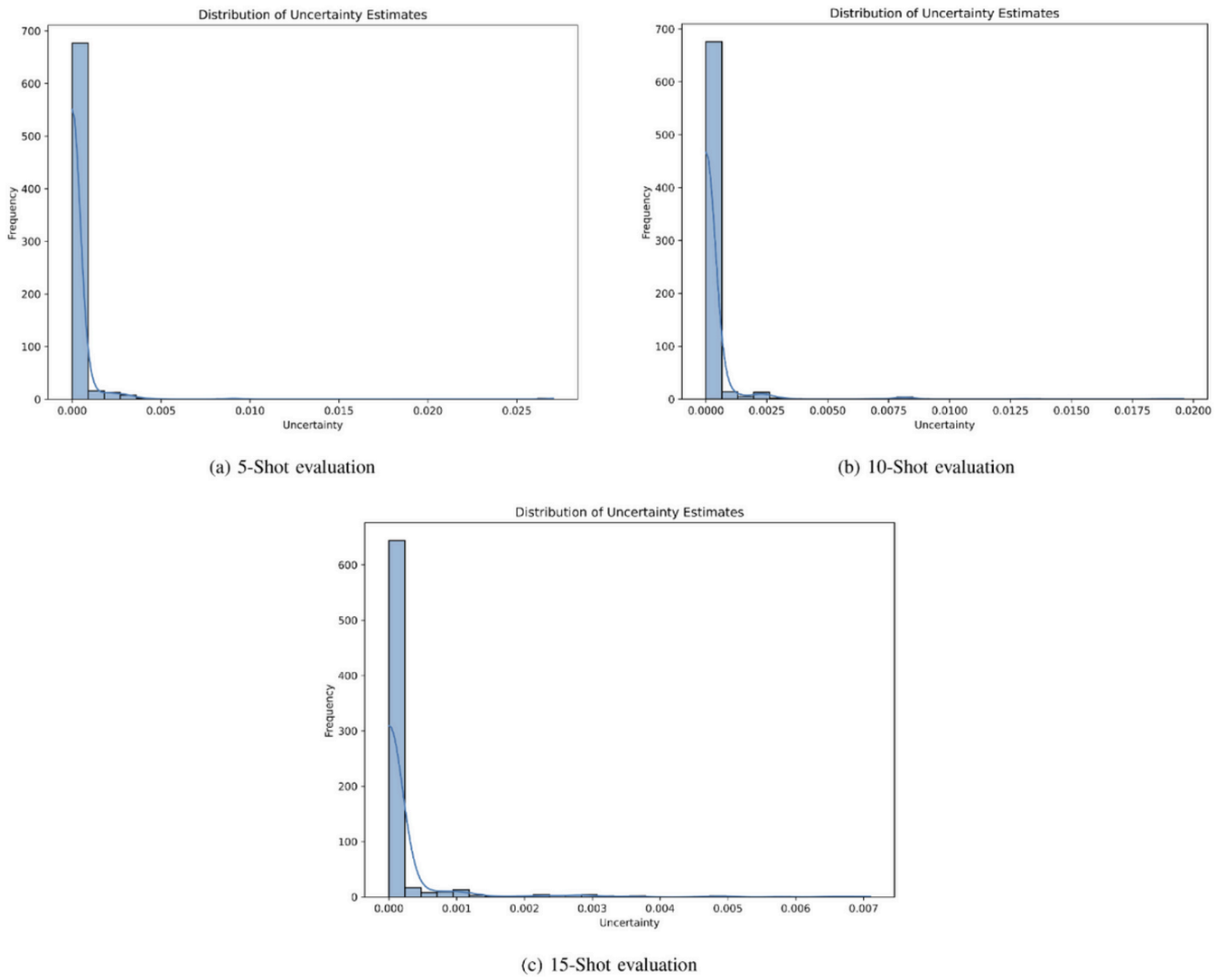
- CPU → data-driven optimization effect
- GPU → predictable hardware requirements for deployment

05 Experiments and Results

D. UNCERTAINTY QUANTIFICATION ANALYSIS

- Extracting a numerical representation of how confident the model is in its predictions
- Enables the automatic flagging of uncertain detections

<"Overall" distribution of uncertainty values>



(a) 5-Shot evaluation

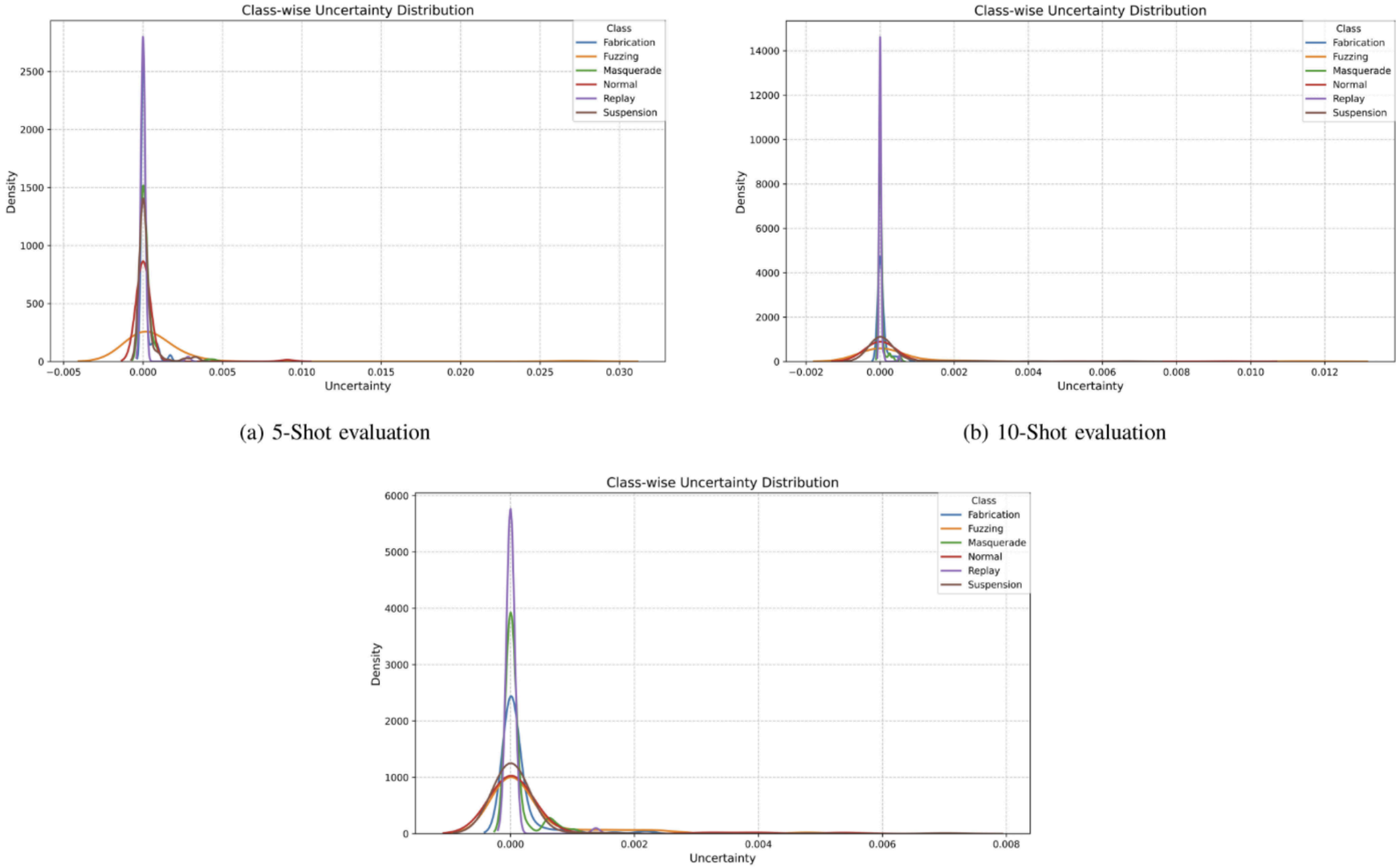
(b) 10-Shot evaluation

(c) 15-Shot evaluation

FIGURE 10. **Distribution of uncertainty estimate for different k-shot evaluations.**

05 Experiments and Results

<Uncertainty distribution by class>



(a) 5-Shot evaluation

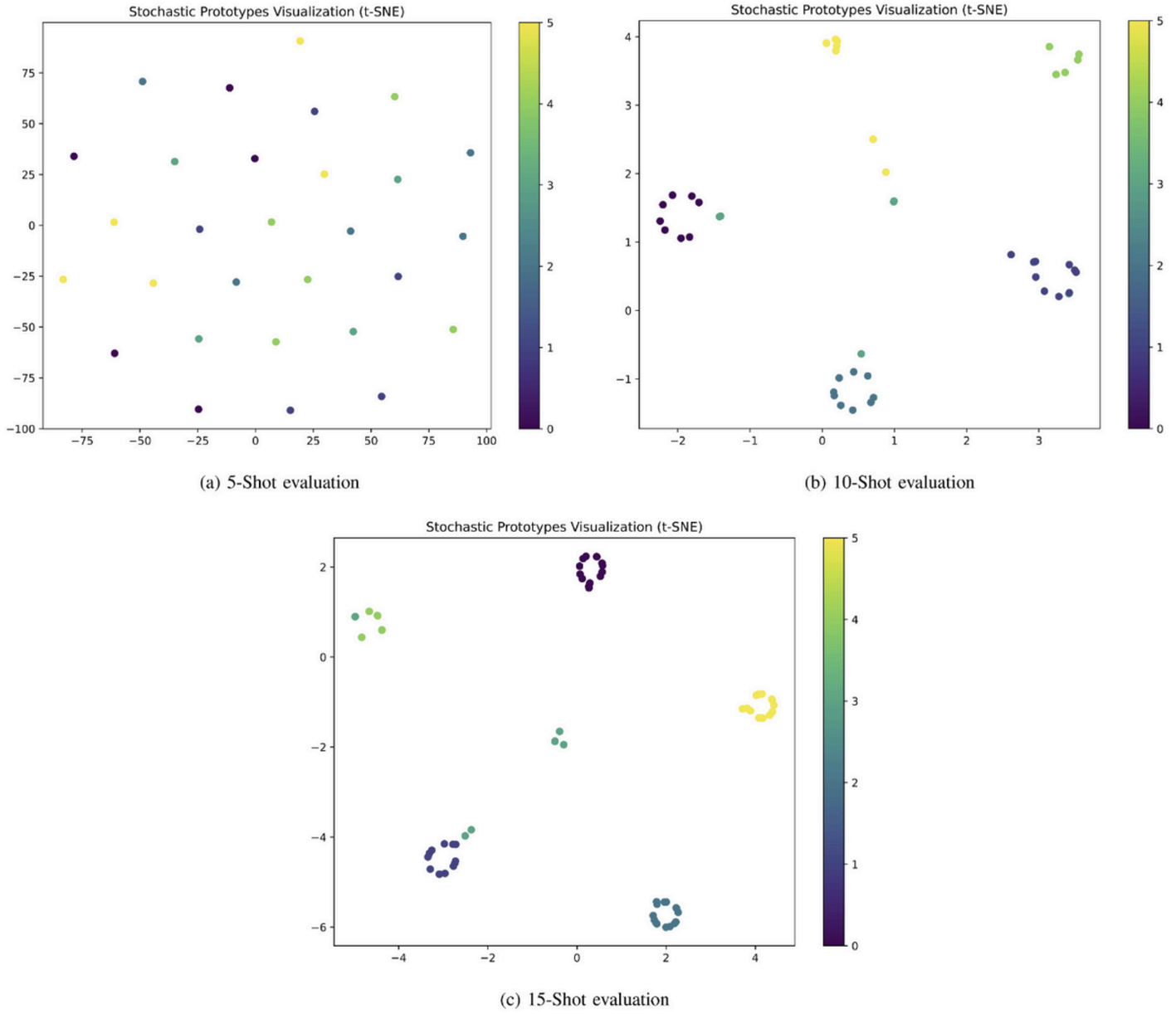
(b) 10-Shot evaluation

(c) 15-Shot evaluation

FIGURE 11. **Class-wise uncertainty distribution for different k-shot evaluations.**

05 Experiments and Results

E. FEATURE SEPARABILITY ANALYSIS VIA T-SNE VISUALIZATIONS



(a) 5-Shot evaluation

(b) 10-Shot evaluation

(c) 15-Shot evaluation

- Persistence of some minor overlap in certain regions even at 15-shot….?
  - May indicate fundamental feature similarity between specific attack types

FIGURE 12. **Stochastic prototype visualizations (t-SNE) for different k-shot evaluations.**

05 Experiments and Results

F. PERFORMANCE COMPARISON WITH BASELINE MODELS

TABLE 2. Performance comparison with baseline FSL methods.

FSL Approach	FSL Setting		Performance Evaluation Metrics				
	N-way	K-shots	Accuracy	Precision	Recall	F1 Score	Training Time (sec)
Siamese Network	5	5	0.8540	0.8364	0.8540	0.8284	121.36
	5	10	0.9155	0.9154	0.9155	0.9084	201.35
	5	15	0.9349	0.9375	0.9349	0.9313	330.50
Relation Network	5	5	0.9367	0.9441	0.9367	0.9313	257.15
	5	10	0.9299	0.9387	0.9299	0.9249	416.34
	5	15	0.9059	0.9093	0.9059	0.8917	581.19
Proposed	5	5	0.9750	0.9762	0.9750	0.9752	44.97
	5	10	0.9833	0.9841	0.9833	0.9831	44.36
	5	15	0.9917	0.9921	0.9917	0.9917	45.91

06 Conclusion

- **Addressed key challenges**
  - Data scarcity, variability in network traffic, dynamic nature of evolving cyber threats
- **Probabilistic nature of RaNNs**
  - enhances the system’s ability to quantify uncertainty
- Demonstrated exceptional classification performance across various FSL scenarios
- Maintained impressive **computational efficiency**
  - millisecond inference times that make it well-suited for real-time deployment
- Ability to **quantify uncertainty** enables it to flag ambiguous cases for further investigation
  - reducing the likelihood of both false positives and negatives

07 Strengths & Limitations

1. Idea of converting distance into **‘confidence’** and making predictions
   - → naturally aligns with the policy of holding and reconfirming ambiguous cases in operational settings
   - May be practical in fields like IoV, where the costs of FP & FN are high
2. Representing classes as **Gaussian distributions** rather than ‘points’
   - → Robust to small amounts of data
3. **A prototype can be created immediately using just K-shots**
   - → Ready for deployment without waiting for retraining
4. The model’s **small size** and **low inference latency**
   - → Ideal for constrained environments like ECUs and gateways

07 Strengths & Limitations

1. The breadth of **comparisons and ablation**
2. The breadth of **evaluation criteria & dataset**
    - Accuracy/confusion matrix/F1 are mainly reported in this paper…
    - But ROC, PR-AUC, average detection latency, etc. may be also important in actual operational IDS
3. **Hyperparameter sensitivity**
    - UQ (uncertainty) quality and running time are sensitive to the number of MC dropouts T, and the choice of α can determine overconfidence/underconfidence in new classes.
4. **Uncertainty Quantification(UQ) part**
    - The specific procedures for ‘further analysis’ are not detailed (ex. human verification, rule-based secondary detection debugging pipelines, etc.)
5. The written structure of the algorithm
6. Unsupervised Learning…? Few-shot vs Zero-shot….?

The End.