Secure On-Device Video OOD Detection Without Backpropagation

Shawn Li[1], Peilin Cai[1], Yuxiao Zhou[2], Zhiyu Ni[3], Renjie Liang[4],
You Qin[2], Yi Nian[1], Zhengzhong Tu[5], Xiyang Hu[6], Yue Zhao[1]
[1]University of Southern California [2]National University of Singapore
[3]University of California, Berkeley [4]University of Florida
[5]Texas A&M University [6]Arizona State University
{li.li02, peilinca, yinian, yzhao010}@usc.edu,
{e1011019, e0962995}@nus.edu.sg,
zhiyuni@berkeley.edu, liang.renjie@ufl.edu,
tzz@tamu.edu, xiyang.hu@asu.edu

Abstract

Out-of-Distribution (OOD) detection is critical for ensuring the reliability of machine learning models in safety-critical applications such as autonomous driving and medical diagnosis. While deploying personalized OOD detection directly on edge devices is desirable, it remains challenging due to large model sizes and the computational infeasibility of on-device training. Federated learning partially addresses this but still requires gradient computation and backpropagation, exceeding the capabilities of many edge devices. To overcome these challenges, we propose ***SecDOOD****, a secure cloud-device collaboration framework for efficient on-device OOD detection without requiring device-side backpropagation. SecDOOD utilizes cloud resources for model training while ensuring user data privacy by retaining sensitive information on-device. Central to SecDOOD is a HyperNetwork-based personalized parameter generation module, which adapts cloud-trained models to device-specific distributions by dynamically generating local weight adjustments, effectively combining central and local information without local fine-tuning. Additionally, our dynamic feature sampling and encryption strategy selectively encrypts only the most informative feature channels, largely reducing encryption overhead without compromising detection performance. Extensive experiments across multiple datasets and OOD scenarios demonstrate that SecDOOD achieves performance comparable to fully fine-tuned models, enabling secure, efficient, and personalized OOD detection on resource-limited edge devices. To enhance accessibility and reproducibility, our code is publicly available at* https://github.com/Dystopians/SecDOOD.

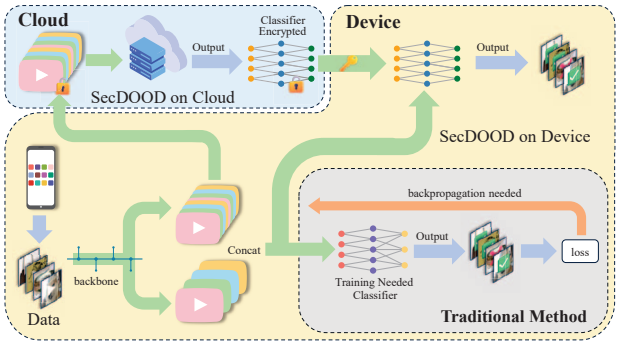

Figure 1. Comparison of traditional OOD detection methods and our proposed SecDOOD. Conventional approaches require training or fine-tuning the classifier/backbone model for OOD detection, which is computationally demanding and challenging to execute on resource-constrained devices. In contrast, SecDOOD introduces a hypernetwork-based solution that eliminates the need for backpropagation on the device, enabling efficient and lightweight OOD detection.

1. Introduction

Out-of-Distribution (OOD) detection [3, 12, 15, 24, 28, 36] is critical for ensuring the reliability of machine learning systems deployed in real-world scenarios [4, 11, 14, 26, 27, 41, 43, 46]. It involves distinguishing in-distribution (ID) samples from unknown OOD samples, which exhibit semantic shifts and unexpected characteristics. Effective OOD detection is critical in safety-sensitive domains, e.g., autonomous driving [8, 25] and medical imaging [19], where incorrect identification can result in severe outcomes.

Need for real-time OOD Detection on Edge Devices. In practical scenarios, OOD detection often involves deployment directly on *edge devices*, such as autonomous vehicles or medical diagnostic equipment, to accommodate user-

specific data distributions and real-time detection requirements. However, deploying OOD detection directly on edge devices faces significant practical challenges. *First*, modern OOD detection models often have large parameter sizes [10, 15], making deployment infeasible on resource-limited edge hardware. *Second*, conventional on-device training or fine-tuning is computationally prohibitive, given limited hardware capabilities. Federated learning [5, 37] addresses data privacy concerns but still requires gradient computation and backpropagation, which exceed the capacity of many edge devices.

To address these challenges, we present **SecDOOD** (**Sec**ure On-**D**evice **OOD** Detection), a cloud-device collaboration approach for secure and efficient OOD detection on edge devices that does *not* require local backpropagation. SecDOOD leverages cloud-based training while ensuring that sensitive user data remains protected on-device during inference. Central to our method is a HyperNetwork-based [13] personalized parameter generation module, which tailors cloud-trained models to device-specific data distributions without on-device fine-tuning. To further improve efficiency, we introduce a dynamic feature sampling and encryption procedure that selectively encrypts only the most informative feature channels before transmission, minimizing encryption overhead while retaining model effectiveness. This design enables privacy-preserving OOD detection that is suitable for resource-constrained edge hardware.

In summary, our contributions are as follows:

- **Backpropagation-Free Deployment.** We propose a cloud-device collaboration way for on-device OOD detection *without* local backpropagation by hypernetworks, enabling deployment on resource-limited edge devices.
- **Personalized Privacy Preservation.** SecDOOD incorporates personalized parameter generation, dynamic feature sampling, and selective encryption to safeguard user data and maintain detection performance.
- **Robust and Efficient Performance.** Through comprehensive evaluations on multiple datasets and OOD tasks, we demonstrate that SecDOOD achieves performance comparable to fully fine-tuned models.

2. Related Works

Out-of-Distribution Detection. The task of Out-of-Distribution (OOD) detection involves identifying samples that deviate from the training distribution while ensuring that in-distribution (ID) classification remains accurate. Methods for OOD detection generally fall into two categories: post hoc techniques and training-time regularization [44]. Post hoc methods analyze model outputs to compute OOD scores, as seen in approaches like Maximum Softmax Probability (MSP) [15]. Enhancements to these methods include temperature scaling and input perturbation techniques [29], while later improvements such as MaxLogit [16] and energy-based scoring [31] have refined detection accuracy. Feature-space modifications, including activation-trimming strategies like ReAct [39] and ASH [7], as well as distance-based classifiers such as Mahalanobis [22] and k-Nearest Neighbor (kNN) [40], leverage learned feature representations for OOD detection. Hybrid models have emerged, such as VIM [42] and Generalized Entropy (GEN) [32], which incorporate both feature statistics and logits to improve detection robustness.

Cloud-Device Collaborative Learning. Cloud-device collaborative learning has gained traction as a means of optimizing computational efficiency while maintaining data privacy [18, 33, 34, 45]. A common implementation is federated learning (FL) [5, 37], where distributed edge devices collaboratively train a shared global model without transmitting raw data to a central server. While FL preserves privacy, it incurs high communication costs due to frequent model synchronization, making it less suitable for large-scale, real-time applications. Alternative paradigms such as split learning [30] and offloading-based computation [23] distribute model processing between cloud servers and edge devices, either by delegating partial training to edge devices or by performing local inference while relying on cloud-based model updates. Additionally, knowledge distillation [45] has been explored as a strategy for transferring pre-trained cloud model knowledge to lightweight device-side models, though such methods often require periodic fine-tuning, imposing computational overhead on resource-limited devices. In contrast, our approach eliminates the need for local model adaptation, training, or fine-tuning. Instead, we employ cloud-assisted data selection and domain adaptation to personalize model outputs while maintaining minimal computational demands on the device. This framework significantly enhances scalability and adaptability, making it a viable solution for large-scale, resource-constrained environments.

3. Proposed Method

3.1. Problem Statement and Preliminaries

Given a training set $\mathbf{D} = \{(x_i, y_i)\}_{i=1}^{n}$, where each $x_i \in \mathcal{X}$ represents an input sample and $y_i \in \mathcal{Y} = \{1, 2, \ldots, C\}$ denotes its corresponding class label, the goal of OOD detection is to differentiate between ID and OOD samples. OOD samples exhibit *semantic shifts* relative to ID samples and do not belong to any predefined class in $\mathcal{Y}$. The model consists of a feature extractor $g(\cdot)$, which derives feature representations, and a classifier $h(\cdot)$, which generates predictions, producing a probability output $\hat{p}$.

The ID data follow a marginal distribution P_{in}, while OOD samples encountered during testing originate from a different marginal distribution P_{out}. The objective of OOD detection is to learn a decision function G that determines

whether a test sample $x \in \mathcal{X}$ belongs to the ID distribution or is an OOD sample:

$$G(x; g, h) = \begin{cases} 0 & \text{if } x \sim \mathcal{D}_{\text{out}}, \\ 1 & \text{if } x \sim \mathcal{D}_{\text{in}}. \end{cases} \tag{1}$$

Problem 1 (On-Device OOD Detection) *In this work, we consider a setting where the OOD detection model is deployed on a device. The model is expected to use only the device's computational resources during the prediction phase in order to protect user data privacy. Let $\mathcal{C}_{device}$ denote the limited computational capacity of the device. In this setting, for any test sample $x \in \mathcal{X}$, the prediction function $G(x; g, h)$ must be evaluated under the constraint $\mathcal{C}(x) \leq \mathcal{C}_{device}$, where $\mathcal{C}(x)$ represents the computational cost associated with processing x.*

Key Challenges in On-Device OOD Detection. One challenge is that the device's computational capacity, denoted by $\mathcal{C}_{\text{device}}$, is limited, making heavy computations (e.g., backpropagation) infeasible on-device. Consequently, the training phase must occur offline, and the device performs only inference with a fixed model, potentially degrading the performance of $G(x; g, h)$ on unseen OOD samples. These challenges imply constraints such as $\mathcal{C}(\theta) \leq \mathcal{C}_{\text{device}}$ and $G(x; g, h)|_{x \sim P_{\text{out}}} \leq \epsilon$, where ϵ indicates the acceptable detection performance.

3.2. Overview

To address the problems and challenges above, it is key to design a collaboration system that leverages cloud resources for comprehensive training and processing, while preserving data privacy and meeting computational constraints. In this system, a test sample is first encrypted on the device and then uploaded to the cloud, where the encrypted real-time test samples $En(\mathcal{S}_T)$ are processed. The processed results are then returned to the device for decryption. In particular, our approach trains a cloud model $\mathcal{M}_g(\cdot; \Theta_g)$ using the personal ID samples $\mathcal{S}_{ID}$ and transfers its knowledge to a local device model $\mathcal{M}_c(\cdot; \Theta_c)$ to ensure that the device, operating under limited computational capacity $\mathcal{C}_{\text{device}}$, maintains acceptable OOD detection performance.

$$\textbf{SecDOOD} : \underbrace{\mathcal{M}_c(\mathcal{S}_{ID}; \Theta_c)}_{\text{Cloud Model}} \rightarrow \underbrace{\mathcal{M}_d(En(\mathcal{S}_T); \Theta_d)}_{\text{Device Model}}. \tag{2}$$

Framework Pipeline. Our framework operates in three distinct stages. In the first stage, the cloud model $\mathcal{M}_g(\cdot)$ is trained using the ID samples $\mathcal{S}_{ID}$, incorporating a HyperNetwork-based module that generates personalized parameters. In the second stage, the device extracts features F_T from its real-time samples $\mathcal{S}_T$ using a pre-trained vision backbone, which remains fixed during the process. The extracted features are then encrypted and uploaded to the cloud. The cloud model $\mathcal{M}_g(\cdot)$ processes these encrypted features to derive personalized parameters for the device model. In the final stage, the device receives and decrypts the updated parameters from the cloud, applies them to its local model $\mathcal{M}_d(\cdot)$, and produces the final predictions.

3.3. Personalized Parameters Generation

Motivation. Adapting the cloud model to real-time samples on a personalized device usually requires on-device fine-tuning, which is difficult due to the computation constraint. To overcome this issue, we propose a HyperNetwork-based [13] personalized parameters generation module implemented as a cloud service. This module processes videos captured by the device and generates device-specific parameters that are designed to adapt the model to the unique data distributions present on the device.

We leverage a HyperNetwork-based approach to generate personalized parameters. Given the cloud model $\mathcal{M}_g(\cdot; \Theta_g)$ trained on ID samples $\mathcal{S}_{ID}$, we define a HyperNetwork $H(\cdot; \Theta_H)$ that learns a mapping from the device-extracted features F_T to personalized model parameters Θ_d for the device model $\mathcal{M}_d(\cdot; \Theta_d)$. Formally, the parameter adaptation process can be written as:

$$\Theta_d = H(F_T; \Theta_H). \tag{3}$$

Here, $H(\cdot; \Theta_H)$ is designed to generate device-specific parameters dynamically, ensuring the model can adapt to heterogeneous data distributions across different devices without requiring direct on-device updates. Instead of learning a single fixed set of parameters, the HyperNetwork learns to generalize across different device conditions, which aligns with the principles of meta-learning—learning how to adapt efficiently to new data distributions.

Meta-Learning Objective. Since device data distributions vary, directly applying a globally trained model often results in performance degradation. To ensure effective adaptation, we formulate the learning of $H(\cdot; \Theta_H)$ as a meta-learning problem [17], where the objective is to optimize the HyperNetwork such that the generated parameters Θ_d allow the device model to perform well on real-time samples. Given an ID dataset $\mathcal{S}_{ID}$ used to train the cloud model and a device-specific real-time dataset $\mathcal{S}_T$, the optimization is formulated as:

$$\min_{\Theta_H} \mathbb{E}_{x \sim \mathcal{S}_T} \mathcal{L}(G(x; g, h_{\Theta_d})), \text{where} \quad \Theta_d = H(F_T; \Theta_H). \tag{4}$$

Here, $\mathcal{L}$ represents the loss function for the OOD detection decision function $G(x; g, h_{\Theta_d})$, ensuring that the generated parameters Θ_d effectively adapt to the local data distribution without requiring on-device fine-tuning. This meta-learning formulation enables the HyperNetwork to gener-

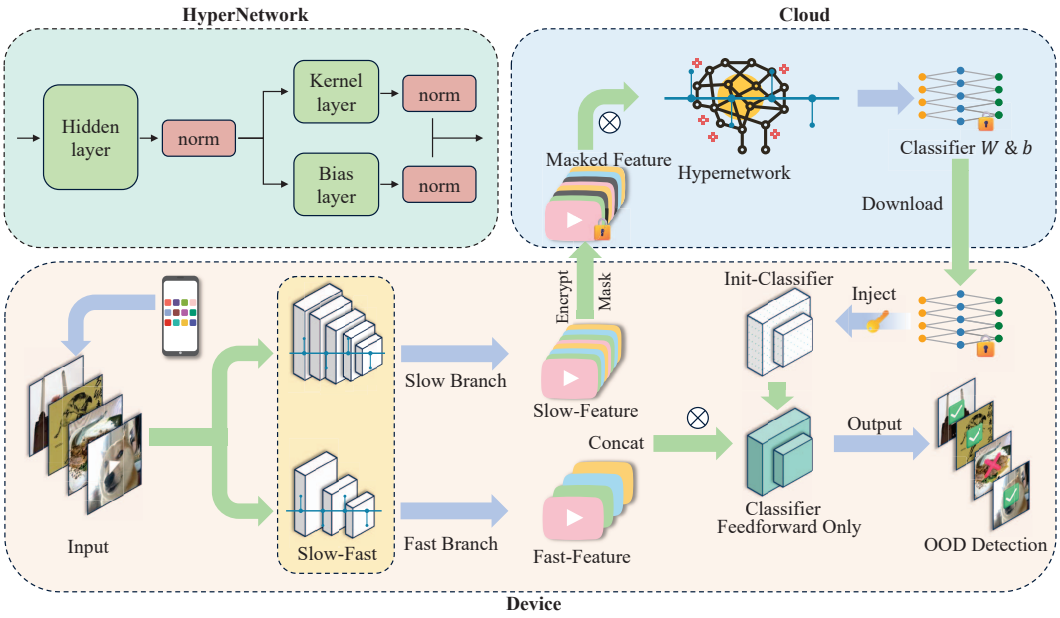

Figure 2. Overview of the proposed SecDOOD framework. SecDOOD leverages cloud-device collaboration to enable efficient and privacy-preserving OOD detection on edge devices. During deployment, the edge device extracts feature representations from incoming data and applies dynamic feature sampling to select the most informative channels. These selected features are securely encrypted and transmitted to the cloud, where a HyperNetwork-based module generates personalized model parameters tailored to the device-specific data distribution. The personalized parameters are then sent back to the device and encrypted, where they are decrypted and used for inference. This approach ensures adaptability to user-specific distributions while maintaining computational efficiency and data privacy.

alize its parameter generation ability across diverse device environments, making it robust to unseen data variations.

Cloud-Device Parameter Transfer. Once the HyperNetwork generates personalized parameters Θ_d, they are securely transmitted to the device for model adaptation. The device model $\mathcal{M}_d(\cdot;\Theta_d)$ applies these updated parameters to perform inference on real-time test samples. Since the device does not perform backpropagation or fine-tuning, this approach efficiently adapts the model while maintaining computational feasibility within the device's resource constraints $\mathcal{C}_{\text{device}}$. By leveraging HyperNetworks with a meta-learning objective, the proposed framework efficiently personalizes OOD detection models for heterogeneous device environments. This enables the system to achieve high detection performance without requiring computationally expensive on-device updates.

However, the parameter generation process relies on features uploaded from the device, which poses a potential risk of user privacy leakage. To mitigate this, we introduce a dynamic feature sampling and encryption module that selectively encrypts critical feature components before transmission. This design ensures that user privacy is protected while preserving the functionality of the HyperNetwork for effective parameter generation.

3.4. Dynamic Feature Sampling and Encryption

Motivation. In order to prevent user privacy leakage, all features to be uploaded to the cloud must be encrypted on the device. However, encryption is computationally expensive due to both the inherent cost of encryption algorithms and the limited processing power of the device. Directly encrypting the entire feature tensor $F_T \in \mathbb{R}^{C\times T\times W\times H}$ would introduce significant latency, making real-time OOD detection infeasible. To address this challenge, we propose the *Dynamic Feature Sampling and Encryption* method, which selectively encrypts only the most critical feature channels while masking less informative ones. This reduces encryption time while maintaining the performance of the personalized parameter generation module.

Channel Importance Estimation. To determine the most influential channels in feature extraction, we employ a game-theoretic importance estimation method inspired by Shapley Additive Explanations [2, 35]. Given a vision backbone $g(\cdot)$ that extracts feature maps F_T from input samples x, our goal is to quantify the contribution of each channel F_T^c, where F_T^c represents the feature tensor corresponding to channel c, to the overall feature representation.

The importance score $S(c)$ of each channel c is computed using the Shapley value formulation:

$$S(c) = \sum_{\mathcal{U} \subseteq \mathcal{C} \setminus \{c\}} \frac{|\mathcal{U}|!(|\mathcal{C}| - |\mathcal{U}| - 1)!}{|\mathcal{C}|!} \left[f(\mathcal{U} \cup \{c\}) - f(\mathcal{U}) \right], \tag{5}$$

where $\mathcal{C} = \{1, 2, \ldots, C\}$ represents the set of all feature channels, and C denotes the total number of channels in F_T. The subset $\mathcal{U}$ includes a selection of channels excluding c, while $f(\mathcal{U})$ quantifies the feature importance when only channels in $\mathcal{U}$ are considered. The term $\frac{|\mathcal{U}|!(|\mathcal{C}|-|\mathcal{U}|-1)!}{|\mathcal{C}|!}$ is a combinatorial coefficient that assigns a weight to each subset $\mathcal{U}$, ensuring a fair contribution estimation for channel c. Here, the factorial notation $n!$ represents the product of all positive integers up to n.

Since computing the exact Shapley values is intractable for large C, we approximate them using Monte Carlo sampling. This involves iteratively estimating $S(c)$ by randomly selecting subsets $\mathcal{U}$ and measuring the marginal contribution of each channel. This approach enables efficient identification of the most important channels for encryption while maintaining computational feasibility.

Dynamic Encryption and Masking Strategy. To reduce encryption overhead while preserving essential information, we rank all channels based on their importance scores $S(c)$ and selectively encrypt the top α fraction, while masking the rest. Our experiments indicate that even when the HyperNetwork receives only a subset of the feature information, it maintains strong generalization capability, effectively generating personalized and parameters for the on-device model.

$$\mathcal{C}_{\text{enc}} = \{c \mid S(c) \text{ in top } \alpha C\}, \quad \mathcal{C}_{\text{mask}} = \mathcal{C}_{\text{all}} \setminus \mathcal{C}_{\text{enc}}. \tag{6}$$

Setting $\alpha = 0.50$, we encrypt only the most informative 50% of channels, denoted as:

$$F_T^{\text{enc}} = \text{Encrypt}(F_T^c), \quad c \in \mathcal{C}_{\text{enc}}, \tag{7}$$

while the remaining 50% are masked, leading to the following structured feature representation:

$$F_T^c = \begin{cases} \text{Encrypt}(F_T^c), & c \in \mathcal{C}_{\text{enc}}, \\ 0, & c \in \mathcal{C}_{\text{mask}}. \end{cases} \tag{8}$$

This strategy ensures that only the most informative channels are encrypted, while the rest are masked to reduce computational overhead. We employ *homomorphic encryption* [1], which enables computation directly on encrypted data. This allows the encrypted feature subset F_T^{enc} to be transmitted to the cloud and processed by the $H(\cdot; \Theta_H)$ without requiring decryption. The generated parameters Θ_d are then securely transmitted back to the device, where they are decrypted and applied for final prediction.

By encrypting only the most critical channels and leveraging homomorphic encryption, this method effectively balances privacy protection, computational efficiency, and model adaptability.

4. Experiment

4.1. Datasets and Evaluation Metrics

Dataset. we evaluate our method across five datasets: HMDB51 [21], UCF101 [38], Kinetics-600 [20], HAC [9], and EPIC-Kitchens [6]. More details on these datasets can be found in Appx. **??**.

Evaluation Metrics. We evaluate the performance via the use of the following metrics: (1) the false positive rate (FPR95, the lower, the better) of OOD samples when the true positive rate of ID samples is at 95%, (2) the area under the receiver operating characteristic curve (AUROC).

4.2. Tasks, Baseline Designs, and Implementation Details

Tasks. We evaluate SecDOOD on two tasks: Near-OOD detection and Far-OOD detection [10]. See details in Appx. **??**

Baseline Design. As our work represents the first attempt to address the OOD detection problem in a real-world deployment scenario, there are no established baseline models for direct comparison. Therefore, we evaluate SecDOOD against conventional baseline methods that require fully training or fine-tuning both a classifier and a visual backbone on the device. Additionally, to further demonstrate the effectiveness of SecDOOD, we design two alternative baselines that, like our approach, do not involve on-device training. See details in Appx. **??**.

Implementation Details and Hardware. In both the Near-OOD and Far-OOD tasks, the batch size is set to 16, the optimizer used is Adam, and the learning rate is set to 0.0001. See more details on implementation and hardware in Appx. **??**.

4.3. Performance and Generalization

Tables 1, 2 and 3 consistently show the effectiveness of SecDOOD in comparison to on-device training methods on 2 tasks with all 5 datasets. SecDOOD consistently achieves the best or near-best values for key metrics, including FPR95 and AUROC.

SecDOOD Excels in Real-World Scenarios. In real-world deployment, device users must effectively detect OOD data originating from diverse distributions. To simulate these conditions, we design Far-OOD detection experiments, where a single dataset is used as the ID data for training (representing users' in-distribution samples), while four additional datasets serve as OOD data (representing diverse unseen distributions). As presented in Tab. 1 and

Methods	HMDB51		UCF101		EPIC-Kitchens		HAC		Average	
	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
With On-Device Training										
MSP	66.83	75.64	67.32	71.13	43.37	86.66	56.17	79.50	58.42	78.23
Energy	72.64	71.75	70.12	71.49	43.66	82.05	61.50	74.99	61.98	75.07
MaxLogit	72.06	73.68	70.47	71.96	39.84	84.76	57.95	77.27	60.08	76.92
ReAct	82.17	67.09	78.44	67.64	53.49	78.07	75.11	70.04	72.80	70.71
ASH	71.62	76.66	69.36	72.38	34.38	88.05	47.85	83.49	55.80	80.15
GEN	68.47	78.43	64.80	73.97	36.81	85.11	49.53	83.67	54.90	80.30
KNN	71.08	78.84	68.62	74.33	41.83	82.32	57.00	82.53	59.63	79.51
VIM	72.25	71.88	70.72	70.58	43.14	82.69	59.48	75.46	61.40	75.15
LogitNorm	66.48	79.08	63.79	75.10	39.03	85.27	54.22	81.83	55.88	80.30
Without On-Device Training										
Ini-Classifier	84.89	66.19	85.43	62.65	87.18	57.94	89.03	63.42	86.63	62.55
Ini-Hypernetwork	86.01	67.31	84.61	63.05	88.86	57.86	88.71	56.72	87.05	61.24
SecDOOD	69.52	79.25	69.34	75.37	31.37	89.01	58.32	84.98	57.14	82.15

Table 1. Far-OOD Detection results using Kinetics-600 as the ID dataset (↑ higher is better; ↓ lower is better). Ini-Classifier refers to a randomly initialized classifier used directly for inference, while Ini-Hypernetwork denotes a randomly initialized hypernetwork that generates parameters for the classifier.

Methods	Kinetics-600		UCF101		EPIC-Kitchens		HAC		Average	
	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
With On-Device Training										
MSP	39.11	88.78	46.64	86.40	17.33	95.99	39.91	89.10	35.75	90.07
Energy	32.95	92.48	44.93	87.95	8.10	97.70	32.95	92.28	29.73	92.60
MaxLogit	33.07	92.31	44.93	88.02	9.12	97.77	33.06	92.17	30.05	92.57
ReAct	27.59	93.54	44.01	88.05	7.53	97.61	31.01	92.86	27.54	93.02
ASH	51.20	87.81	53.93	84.22	19.95	95.92	42.99	90.23	42.02	89.55
GEN	41.51	90.34	46.18	87.91	8.21	98.26	38.31	91.28	33.55	91.95
KNN	22.69	95.01	39.34	89.28	9.92	97.92	20.75	96.02	23.18	95.06
VIM	13.68	97.01	33.87	91.45	5.93	98.15	13.45	97.12	16.73	93.93
LogitNorm	46.07	87.41	49.03	85.96	15.96	96.30	47.09	87.64	39.54	89.33
Without On-Device Training										
Ini-Classifier	94.18	43.60	96.58	40.51	97.49	41.39	85.40	59.71	93.41	46.30
Ini-Hypernetwork	89.97	48.99	86.77	53.69	90.36	57.22	92.13	57.28	89.81	54.30
SecDOOD	15.39	96.06	46.18	86.19	5.01	98.72	22.35	94.41	22.23	93.85

Table 2. Far-OOD Detection results using HMDB as the ID dataset (↑ higher is better; ↓ lower is better). Ini-Classifier refers to a randomly initialized classifier used directly for inference, while Ini-Hypernetwork denotes a randomly initialized hypernetwork that generates parameters for the classifier.

Tab. 2, SecDOOD significantly outperforms other 'Without On-Device Training' baseline methods, demonstrating its effectiveness. In addition, SecDOOD achieves superior results compared to most 'With On-Device Training' methods. Notably, on the EPIC-Kitchens dataset, SecDOOD surpasses the best 'With On-Device Training' method by +5.12 in AUROC and -10.36 in FPR95 average. This performance not only highlights the effectiveness of SecDOOD in real-world OOD detection but also underscores the strength of our training strategy, which enables superior results even compared to fully trained models.

SecDOOD Demonstrates Efficiency Across Different Tasks. Beyond evaluating SecDOOD on far-OOD detection, we also examine its performance on a more challenging near-OOD scenario, where ID and OOD data originate from the same distribution. In this case, the OOD detection model must be highly effective in distinguishing ID from OOD samples due to their inherent similarity. As shown in Tab. 3, SecDOOD consistently outperforms traditional baseline methods that require on-device training. Notably, on the UCF101 dataset, SecDOOD reduces FPR95 by 38.4% compared to the best 'With On-Device Training' method, highlighting its superior capability in handling near-OOD detection.

SecDOOD is Effective Across Diverse Datasets. To validate the effectiveness of SecDOOD, we conduct experi-

Methods	HMDB51		UCF101		EPIC-Kitchens		Kinetics-600	
	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
With On-Device Training								
MSP	44.66	87.74	22.14	95.73	76.31	67.59	64.08	76.16
Energy	43.36	87.46	22.52	96.06	76.68	68.29	68.75	75.49
MaxLogit	43.36	87.75	22.52	96.02	76.68	68.29	68.73	75.98
Mahalanobis	40.31	85.28	12.14	97.14	98.69	42.99	93.51	35.83
ReAct	42.05	87.79	25.63	95.85	83.96	65.89	72.40	73.80
ASH	53.59	87.16	32.14	94.02	76.87	68.52	69.03	75.33
GEN	43.79	87.49	23.79	95.54	75.93	63.60	69.24	76.16
KNN	42.92	88.06	15.63	96.93	77.05	65.60	68.67	74.64
VIM	36.82	88.06	12.52	97.66	80.97	63.41	68.77	75.47
LogitNorm	48.84	87.65	19.61	95.85	80.97	63.41	67.32	75.84
Without On-Device Training								
Baseline Classifier	94.77	52.27	80.58	66.13	87.74	58.30	89.19	57.50
Baseline Hypernetwork	95.21	54.04	81.81	59.69	96.27	46.66	91.29	55.17
SecDOOD	37.03	88.66	7.48	98.39	62.69	74.93	62.36	76.92

Table 3. Comparison of Multimodal Near-OOD detection methods with and without on-device training across multiple datasets (↑ the higher the better; ↓ the lower the better).

Methods	No Mask		Mask 50% Channels		Mask 75% Channels	
	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
HMDB	37.69	88.72	37.03	88.66	45.75	85.86
UCF	11.55	95.81	7.48	98.36	13.69	96.28
EPIC	63.43	75.36	62.69	74.93	63.43	71.56
Kinetics	62.63	76.36	62.36	76.92	63.85	76.56

Table 4. Near-OOD Detection results using various mask proportion (↑ higher is better; ↓ lower is better).

ments on five datasets, each representing distinct distributions, including kitchen environments, YouTube videos, and everyday scenes. Among them, Kinetics-600 stands out as a particularly challenging dataset, comprising 480,000 video samples. Despite the diversity in data distributions, SecDOOD consistently delivers strong performance across all datasets, demonstrating its robustness and adaptability in various real-world scenarios.

4.4. In-Depth Analysis

To gain a deeper understanding of the efficiency and effectiveness of our method, we conduct a comprehensive analysis across multiple aspects of computational performance and model behavior. Specifically, we evaluate: (1) the effect of masking different numbers of channels on overall performance, (2) a FLOPs comparison between the baseline and our proposed model, and (3) the per-sample encryption and decryption time under varying feature channel settings. These analyses are presented in the main paper. Additionally, we examine (4) the communication latency between the device and the cloud under different network conditions, which is provided in the Appx. **??**.

Methods	No Mask		Randomly Mask 50%		Dynamic Mask 50%	
	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
Near-OOD						
HMDB	37.69	88.72	39.22	89.31	37.03	88.66
UCF	11.55	95.81	12.62	96.32	7.48	98.36
EPIC	63.43	75.36	63.42	71.69	62.69	74.93
Kinetics	62.63	76.36	64.22	76.05	62.36	76.92
Far-OOD (HMDB as ID)						
Kinetics	21.89	94.29	16.89	95.90	15.39	96.06
UCF	52.79	81.93	47.54	86.24	46.18	86.19
HAC	29.42	94.02	29.65	93.70	22.35	94.41
Far-OOD (Kinetics as ID)						
HMDB	69.67	78.09	69.84	77.36	69.52	79.25
UCF	69.63	72.98	70.89	72.84	69.34	75.37
HAC	68.01	76.81	69.15	75.38	58.32	84.98

Table 5. OOD Detection results using various mask strategies (↑ higher is better; ↓ lower is better).

Number of Channels Masked. To evaluate the effectiveness of our dynamic feature sampling approach, we conduct experiments on SecDOOD with different feature masking percentages: (1) masking 75% of the channels, (2) masking 50%, and (3) no masking applied, as shown in Tab. 4 and Appx. **??**. Additionally, we introduce a random masking strategy as a baseline to further highlight the advantages of our dynamic masking mechanism. As shown in Tab. 5, an interesting observation is that masking 50% of the channels often yields comparable or even better performance than applying no masking at all. This suggests that our dynamic masking method effectively filters out noisy and less informative features, allowing the model to focus on more rele-

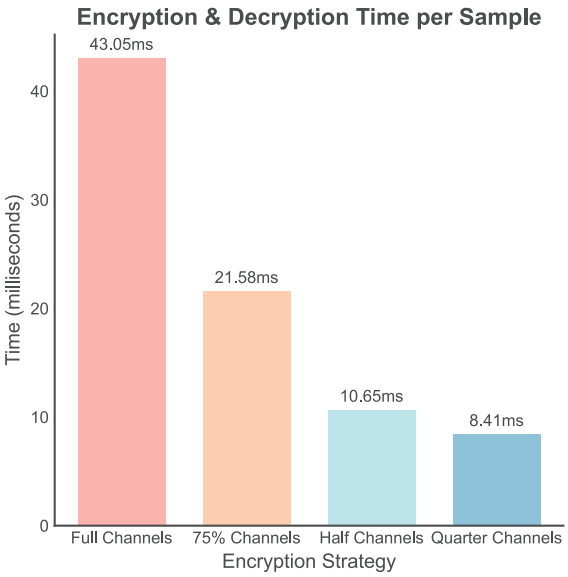

Figure 3. Encryption and decryption time per sample under different feature channel encryption ratios. Encrypting all channels (100%) results in significantly higher time costs, while encrypting 50% and 25% of the channels shows similar computational efficiency. Based on this analysis, we select 50% encryption as the optimal balance between security and efficiency.

vant representations. Furthermore, the results indicate that the HyperNetwork is highly robust to variations in input features, adapting effectively to the refined feature space and maintaining strong detection performance.

Encryption & Decryption Time with Different Feature Channels. We evaluate the encryption and decryption time for a single sample under different feature channel encryption ratios. Specifically, we consider four settings: encrypting 25%, 50%, 75%, and 100% of the channels. As shown in Fig. 3, encrypting all channels (100%) incurs a significantly higher time cost compared to encrypting 50% or 25% of the channels. Notably, the time required for encrypting & decryption 50% and 25% of the channels is relatively similar, suggesting diminishing returns in efficiency gains for lower encryption ratios. Considering both computational efficiency and the impact of different encryption ratios on model performance, we ultimately select the 50% encryption setting as the optimal balance.

FLOPs: Traditional On-Device Training vs. SecDOOD. Traditional on-device training methods require both training a classifier and fine-tuning the visual backbone, resulting in substantial computational overhead. In contrast, SecDOOD offloads the training process to the cloud, eliminating the need for on-device training. Specifically, our approach trains a HyperNetwork in the cloud, which generates personalized classifier parameters while keeping the visual backbone fixed. During deployment, the edge device only performs feature extraction and classifier inference (forward propagation), significantly reducing computational demands. To quantify this efficiency gain, we compare the FLOPs of the traditional training paradigm with SecDOOD. As shown in Fig. 4, SecDOOD reduces computational costs by a factor of three compared to traditional methods that require full on-device training. This reduction underscores the advantage of our cloud-assisted approach, enabling efficient and scalable OOD detection on resource-constrained edge devices. However, when comparing the FLOPs of SecDOOD to baseline inference, we observe a modest increase of 16%, primarily due to encryption and decryption operations on the device.

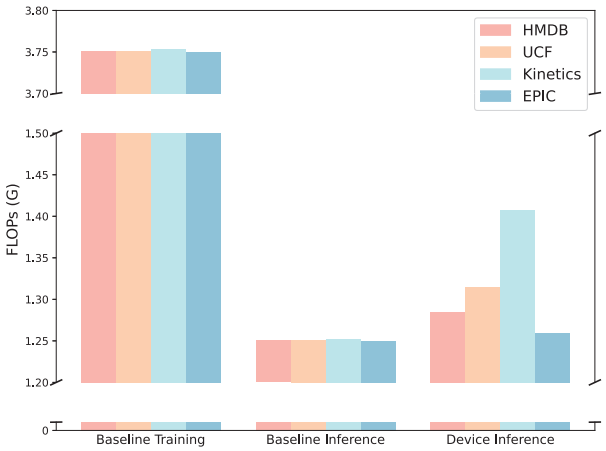

Figure 4. FLOPs comparison between traditional on-device training and SecDOOD. Traditional methods require full model training on the device, incurring high computational costs. SecDOOD offloads training to the cloud, reducing on-device computation to feature extraction and inference, achieving a 3× efficiency gain.

5. Conclusion, Limitations, and Future Work

In this paper, we introduced SecDOOD, a secure and efficient cloud-device collaboration framework for out-of-distribution detection on resource-constrained devices. SecDOOD eliminates the need for backpropagation by leveraging a HyperNetwork-based personalized parameter generation module, enabling effective adaptation to user-specific data distributions. Additionally, the proposed dynamic feature sampling and encryption strategy reduces encryption overhead while preserving model performance, ensuring privacy and efficiency during cloud-device interactions.

Limitations and Future Works. Despite its advantages, SecDOOD has certain limitations. It assumes stable cloud connectivity, which may not always be available in real-world scenarios. Future work will focus on reducing the cost of cloud-device communication, and optimizing the framework for ultra-low-power devices, further broadening its applicability in diverse real-world environments.

Acknowledgments

This work was partially supported by the National Science Foundation under Award Nos. 2428039 and 2346158. The opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors also gratefully acknowledge support from the Amazon Research Awards.

References

[1] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)*, 51(4):1–35, 2018. 5

[2] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. Explaining anomalies detected by autoencoders using shapley additive explanations. *Expert systems with applications*, 186:115736, 2021. 4

[3] Yichen Bai, Zongbo Han, Bing Cao, Xiaoheng Jiang, Qinghua Hu, and Changqing Zhang. Id-like prompt learning for few-shot out-of-distribution detection. In *CVPR*, pages 17480–17489, 2024. 1

[4] Wonwoo Cho, Jeonghoon Park, and Jaegul Choo. Training auxiliary prototypical classifiers for explainable anomaly detection in medical image segmentation. In *WACV*, pages 2624–2633, 2023. 1

[5] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Fedavg with fine tuning: Local updates lead to representation learning. *NeurIPS*, 2022. 2

[6] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018. 5

[7] Andrija Djurisic, Nebojsa Bozanic, Arjun Ashok, and Rosanne Liu. Extremely simple activation shaping for out-of-distribution detection. *arXiv preprint arXiv:2209.09858*, 2022. 2

[8] Hao Dong, Xianjing Zhang, Jintao Xu, Rui Ai, Weihao Gu, Huimin Lu, Juho Kannala, and Xieyuanli Chen. Superfusion: Multilevel lidar-camera fusion for long-range hd map generation. *arXiv preprint arXiv:2211.15656*, 2022. 1

[9] Hao Dong, Ismail Nejjar, Han Sun, Eleni Chatzi, and Olga Fink. SimMMDG: A simple and effective framework for multi-modal domain generalization. In *NeurIPS*, 2023. 5

[10] Hao Dong, Yue Zhao, Eleni Chatzi, and Olga Fink. Multiood: Scaling out-of-distribution detection for multiple modalities. In *NeurIPS*, 2024. 2, 5

[11] Xuefeng Du, Xin Wang, Gabriel Gozum, and Yixuan Li. Unknown-aware object detection: Learning what you don't know from videos in the wild. In *CVPR*, pages 13678–13688, 2022. 1

[12] Mark S. Graham, Walter H. L. Pinaya, Petru-Daniel Tudosiu, Parashkev Nachev, Sebastien Ourselin, and M. Jorge Cardoso. Denoising diffusion models for out-of-distribution detection. In *CVPRW*, pages 2948–2957, 2023. 1

[13] David Ha, Andrew M Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2022. 2, 3

[14] Nan Hao, Yuangang Li, Kecheng Liu, Songtao Liu, Yingzhou Lu, Bohao Xu, Chenhao Li, Jintai Chen, Ling Yue, Tianfan Fu, et al. Artificial intelligence-aided digital twin design: A systematic review. 2024. 1

[15] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017. 1, 2

[16] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joe Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings. *ICML*, 2022. 2

[17] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021. 3

[18] Wei Ji, Li Li, Zheqi Lv, Wenqiao Zhang, Mengze Li, Zhen Wan, Wenqiang Lei, and Roger Zimmermann. Backpropagation-free multi-modal on-device model adaptation via cloud-device collaboration. *ACM Trans. Multimedia Comput. Commun. Appl.*, 21(2), 2025. 2

[19] Davood Karimi and Ali Gholipour. Improving calibration and out-of-distribution detection in deep models for medical image segmentation. *IEEE transactions on artificial intelligence*, 4(2):383–397, 2022. 1

[20] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 5

[21] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011. 5

[22] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018. 2

[23] Dongbo Li, Yuchen Sun, Jielun Peng, Siyao Cheng, Zhisheng Yin, Nan Cheng, Jie Liu, Zhijun Li, and Chenren Xu. Dual network computation offloading based on drl for satellite-terrestrial integrated networks. *IEEE Transactions on Mobile Computing*, 2024. 2

[24] Jingyao Li, Pengguang Chen, Zexin He, Shaozuo Yu, Shu Liu, and Jiaya Jia. Rethinking out-of-distribution (ood) detection: Masked image modeling is all you need. In *CVPR*, pages 11578–11589, 2023. 1

[25] Jinlong Li, Baolu Li, Zhengzhong Tu, Xinyu Liu, Qing Guo, Felix Juefei-Xu, Runsheng Xu, and Hongkai Yu. Light the night: A multi-condition diffusion framework for unpaired low-light enhancement in autonomous driving. In *CVPR*, pages 15205–15215, 2024. 1

[26] Li Li, Chenwei Wang, You Qin, Wei Ji, and Renjie Liang. Biased-predicate annotation identification via unbiased visual predicate representation. In *ACM MM*, pages 4410–4420, 2023. 1

[27] Li Li, Wei Ji, Yiming Wu, Mengze Li, You Qin, Lina Wei, and Roger Zimmermann. Panoptic scene graph generation

with semantics-prototype learning. In *AAAI*, pages 3145–3153, 2024. 1

[28] Shawn Li, Huixian Gong, Hao Dong, Tiankai Yang, Zhengzhong Tu, and Yue Zhao. Dpu: Dynamic prototype updating for multimodal out-of-distribution detection, 2024. 1

[29] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018. 2

[30] Zheng Lin, Guanqiao Qu, Xianhao Chen, and Kaibin Huang. Split learning in 6g edge networks. *IEEE Wireless Communications*, 2024. 2

[31] Weitang Liu, Xiaoyun Wang, John D Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *NeurIPS*, 2020. 2

[32] Xixi Liu, Yaroslava Lochman, and Christopher Zach. Gen: Pushing the limits of softmax-based out-of-distribution detection. In *CVPR*, 2023. 2

[33] Zheqi Lv, Zhengyu Chen, Shengyu Zhang, Kun Kuang, Wenqiao Zhang, Mengze Li, Beng Chin Ooi, and Fei Wu. Ideal: Toward high-efficiency device-cloud collaborative and dynamic recommendation system. *arXiv preprint arXiv:2302.07335*, 2023. 2

[34] Zheqi Lv, Wenqiao Zhang, Shengyu Zhang, Kun Kuang, Feng Wang, Yongwei Wang, Zhengyu Chen, Tao Shen, Hongxia Yang, Beng Chin Ooi, et al. Duet: A tuning-free device-cloud collaborative parameters generation framework for efficient device model generalization. In *WWW*, pages 3077–3085, 2023. 2

[35] Yasunobu Nohara, Koutarou Matsumoto, Hidehisa Soejima, and Naoki Nakashima. Explanation of machine learning models using improved shapley additive explanation. In *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, pages 546–546, 2019. 4

[36] Yuehan Qin, Yichi Zhang, Yi Nian, Xueying Ding, and Yue Zhao. Metaood: Automatic selection of ood detection models. *arXiv preprint arXiv:2410.03074*, 2024. 1

[37] Fahad Sabah, Yuwen Chen, Zhen Yang, Muhammad Azam, Nadeem Ahmad, and Raheem Sarwar. Model optimization techniques in personalized federated learning: A survey. *Expert Systems with Applications*, 243:122874, 2024. 2

[38] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 5

[39] Yiyou Sun, Chuan Guo, and Yixuan Li. React: Out-of-distribution detection with rectified activations. In *NeurIPS*, 2021. 2

[40] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. *ICML*, 2022. 2

[41] Zhensu Sun, Li Li, Yan Liu, Xiaoning Du, and Li Li. On the importance of building high-quality training datasets for neural code search. In *ICSE*, page 1609–1620, 2022. 1

[42] Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. Vim: Out-of-distribution with virtual-logit matching. In *CVPR*, 2022. 2

[43] Haoyan Xu, Kay Liu, Zhengtao Yao, Philip S Yu, Kaize Ding, and Yue Zhao. Lego-learn: Label-efficient graph open-set learning. *arXiv preprint arXiv:2410.16386*, 2024. 1

[44] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, et al. Openood: Benchmarking generalized out-of-distribution detection. In *NeurIPS*, 2022. 2

[45] Jiangchao Yao, Feng Wang, Kunyang Jia, Bo Han, Jingren Zhou, and Hongxia Yang. Device-cloud collaborative learning for recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3865–3874, 2021. 2

[46] Wenting Yi, Wai Kit Chan, Hiu Hung Lee, Steven T Boles, and Xiaoge Zhang. An uncertainty-aware deep learning model for reliable detection of steel wire rope defects. *IEEE Transactions on Reliability*, 2023. 1