

An Efficient Edge-Cloud Collaboration System With Foundational Models for Open-Set IoT Applications

Bufang Yang^{ID}, Wenrui Lu, Lixing He^{ID}, Neiwen Ling^{ID}, Zhenyu Yan^{ID}, *Member, IEEE*,
Guoliang Xing^{ID}, *Fellow, IEEE*, Xian Shuai, Xiaozhe Ren, and Xin Jiang

Abstract—Artificial intelligence (AI) models have been widely deployed on edge devices, enabling various IoT applications. However, lightweight on-device AI models on resource-limited edge devices hinder their adaptability to dynamic environments and tasks. Despite the superior generalization capabilities of recently developed Foundation Models (FMs), utilizing their extensive knowledge on the resource-constrained edge platforms remains unexplored. In this work, we introduce DeepEdgeFM, an edge-cloud collaborative system with FMs that enables open-set learning, simultaneously achieving generalizability and efficiency for IoT applications. DeepEdgeFM employs a spatiotemporal-aware semantic customization approach that leverages spatial, temporal, and domain-specific knowledge from FMs to continuously customize edge models using unlabeled sensor data in emerging IoT environments. Meanwhile, DeepEdgeFM utilizes a dynamic model switching strategy to selectively query the knowledge of FMs based on sensor-data uncertainty and real-time network fluctuations. We implement DeepEdgeFM on five FMs and multi-modal large language models (MLLMs), covering four types of sensor data modalities. We evaluate DeepEdgeFM on two edge platforms, five public datasets, and two self-collected datasets covering both indoor and outdoor real-world environments. The results show that DeepEdgeFM outperforms state-of-the-art baselines, achieving up to an 18.6% accuracy gain and a 38.6% latency reduction.

Index Terms—Foundation models, multi-modal large language models (MLLMs), edge computing, edge-cloud collaborative system, open-set recognition, Internet of Things.

I. INTRODUCTION

RECENT years have witnessed remarkable advancements in the computational power of edge artificial intelligence (AI) hardware. The market for AI hardware is expected to reach 234.6 billion by 2025 [1]. These advancements have enabled the deployment of various deep learning models on embedded devices, facilitating diverse applications in daily life including

healthcare [2], [3], autonomous driving [4], and smart home systems [5]. Since these applications operate in complex and ever-changing environments, generalization, adaptability, and efficiency are critically important.

Due to the limited resources of edge devices, existing on-device AI models are task-specific and only support the predefined classes, working in a *closed-set* manner [2], [6]. Emerging new tasks or sensor data not predefined in the training stage will significantly impair their performance. Although prior efforts such as transfer learning [7] and meta learning [8] can alleviate these challenges, they still require non-trivial manual labeling and on-device retraining, which is impractical for real-world IoT applications [9]. The emergence of foundation models (FMs)¹ such as GPT-4o [10] and ImageBind [11] have the potential to transform human life with their superior generalization and reasoning capabilities, empowering diverse novel applications such as conversational systems [12], [13], multi-modal assistive systems [14], and IoT sensing [15]. Multi-modal FMs such as ImageBind [11] leverage large-scale paired multi-modal data for pretraining, projecting different modalities, including image, audio, depth, and text, into a unified feature space. This unified embedding space bridges the diverse sensor data modalities with text, enabling FMs with **open-set recognition** capabilities.

However, most existing FMs are pretrained on billions of data samples and comprise billions of parameters [16], [17], making them cloud-centric on service platforms such as Microsoft Azure [18], [19], [20]. Offloading sensor data to the cloud is impractical in many IoT scenarios due to network conditions. Directly executing FMs on edge devices is also infeasible due to their heavy computational requirements. Existing model compression solutions [21], [22], [23] treat all sensor data equally, leading to sub-optimal results and potential performance degradation for unseen data. Although prior works on splitting computing [24], [25], [26] can dynamically adjust computing strategy by splitting the model into different parts, the unique characteristics of Transformer architecture in FMs offer limited benefits, as the intermediate features in Transformers are even larger than the raw sensor data [27].

In this paper, we introduce DeepEdgeFM, a novel edge-cloud collaborative system with open-set capability. As illustrated in Fig. 1, DeepEdgeFM maintain FMs on the cloud and

Received 6 February 2025; revised 22 December 2025; accepted 10 February 2026. Date of publication 26 February 2026; date of current version 5 June 2026. This work was supported in part by the Research Grants Council (RGC) of Hong Kong through General Research Fund under Grant 14207123 and in part by Strategic Topics under Grant STG1/E-403/24-N and Grant T43-513/23-N. Recommended for acceptance by S. Nirjon. (Corresponding authors: Zhenyu Yan; Guoliang Xing.)

Bufang Yang, Wenrui Lu, Lixing He, Neiwen Ling, Zhenyu Yan, and Guoliang Xing are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, SAR, China (e-mail: bfyang@link.cuhk.edu.hk; wenrui.lu@link.cuhk.edu.hk; 1155170464@link.cuhk.edu.hk; lingnw@link.cuhk.edu.hk; zzyan@cuhk.edu.hk; glxing@cuhk.edu.hk).

Xian Shuai, Xiaozhe Ren, and Xin Jiang are with the Noah's Ark Lab in Huawei Technologies, Hong Kong, SAR, China (e-mail: shuaixian1@huawei.com; renxiaozhe@huawei.com; Jiang.Xin@huawei.com).

Digital Object Identifier 10.1109/TMC.2026.3666311

¹Throughout this study, FMs encompass both encoder-based models like ImageBind and multi-modal large language models like GPT-4o.

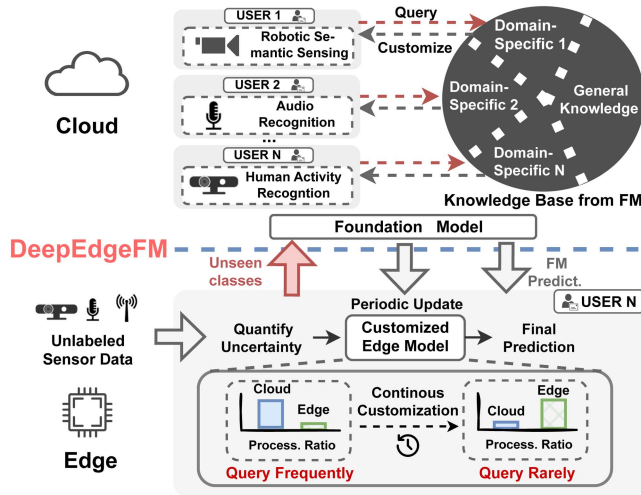


Fig. 1. An example of DeepEdgeFM, leveraging edge-cloud collaborative inference and dynamic customization to enable edge devices with open-set capability.

a customized small model on the edge devices. The FMs on the cloud serve as a knowledge base containing both general and domain-specific knowledge. At the inference stage, DeepEdgeFM integrates the uncertainty of the sensor data and the network conditions to perform dynamic model switching, determining whether to use the customized edge model for efficient inference or FMs for comprehensive inference. Meanwhile, DeepEdgeFM employs a spatiotemporal-aware semantic customization approach to continuously customize the domain-specific knowledge for edge models in a *label-free* manner, gradually enhancing their capabilities and increasing the ratio of data processed on the edge, thereby reducing system overhead.

We implement DeepEdgeFM on five FMs, including encoder-based FMs (CLIP [28], ImageBind [11], LanguageBind [29]) and MLLMs (LLaVA [30] and Qwen-VL [31]), and test it on two edge platforms, NVIDIA AGX Xavier and NVIDIA Nano. We evaluate DeepEdgeFM on five public datasets and two real-world datasets covering both indoor and outdoor environments, using four sensor modalities including RGB, audio, depth, and infrared. Our extensive experiments show that DeepEdgeFM achieves both end-to-end latency reduction and accuracy increase compared to baselines.

In summary, we make the following key contributions:

- We design DeepEdgeFM, an efficient edge-cloud collaborative system with FMs that enables open-set learning for IoT applications. DeepEdgeFM dynamically coordinates cloud-based FMs with lightweight, customized models on the edge to achieve both system generalizability and efficiency.
- We propose a spatiotemporal-aware semantic customization approach that enables DeepEdgeFM to continuously customize the edge model in a *label-free* manner by leveraging the domain-specific knowledge from FMs. This approach enables DeepEdgeFM to efficiently update the edge model and dynamically balance emerging knowledge from FMs with the historical knowledge within the edge model.

- We develop a dynamic model switching strategy that integrates both the uncertainty of sensor data and live network conditions to coordinate FM and customized edge models.
- We implement DeepEdgeFM on five FMs, including both encoder-based FMs and MLLMs, and evaluate it on two edge platforms. We evaluate DeepEdgeFM on five public datasets and two self-collected datasets, covering four sensor data modalities. Results show that DeepEdgeFM achieves up to 3.2x latency reduction and 34.3% accuracy gain compared to the state-of-the-art baselines.

II. RELATED WORK

Open-set Recognition: This paradigm implies that the machine learning model must be capable of identifying any class described in natural language without requiring fine-tuning. Previous studies on open-set recognition can be categorized into semantic-based and generative adversarial network (GAN)-based approaches. Semantic-based solutions [37], [38], [39] establish relationships between language and other modalities in the embedding space. GAN-based solutions [40] employ GANs to synthesize features of unseen classes, using these synthetic features to train the model and adapt to new classes. However, their generalization capabilities remain constrained by limited training data and model size compared to FMs.

Efficient On-device Inference: Various techniques have been proposed to speed up on-device neural network (NN) inference, including model compression [41], early exit [42], input filtering [43], and result reuse [44]. However, model compression techniques such as quantization [41] and pruning [45] are static acceleration solutions, assigning the same computations to all samples, which is suboptimal. Additionally, highly compressed FMs can result in significant performance degradation. Early exit approaches [42] can dynamically reduce redundant computations, but the early-exit heads in FMs incur nontrivial costs due to their deep layers and high-dimensional Transformer embeddings [27]. Further, difficult samples still require the complete inference of the FM, which is impractical on resource-limited edge devices.

Inference Efficiency of FMs: vLLM is a fast LLM serving technique based on PagedAttention [46] and quantization. In addition, techniques such as model cascade [47], speculative decoding [48], layer-skipping [49], and mixture-of-experts (MoE) [34], [50] have also emerged to reduce the inference overhead of FMs. However, these approaches cannot support running FMs on edge devices like the NVIDIA Jetson Nano. Some studies use knowledge distillation (KD) to generate lightweight FMs such as DIME-FM [21] and MobileSAM [23]. However, they primarily aim to preserve open-set capabilities rather than tailor a lightweight edge model, limiting their practicality in embedded systems.

Edge-cloud Collaboration: Several edge-cloud collaboration systems have been proposed in recent years. Neurosurgeon [24] splits the NN into two parts to be deployed on the edge and cloud, respectively. SPINN [25] further integrates an early-exit mechanism into edge-cloud collaboration, jointly optimizing the

TABLE I
A SUMMARY OF RECENT STUDIES ON LEVERAGING FMS ON EDGE DEVICES (● INDICATES INCLUDED)

Approach	Open-set Recognition	Continuous Interaction	LLM-Compatible	Edge Cost Reduction	Modalities
Xue <i>et. al</i> [32]	●	○	○	Low	IMU, WiFi, mmWave
EdgeTA [33]	○	●	○	Medium	RGB, Text
EdgeMoE [34]	○	○	●	Medium	Text
EdgeShard [35]	○	○	●	Medium	Text
PICE [36]	○	○	●	Medium	Text
DeepEdgeFM	●	●	●	High	RGB, Audio, Depth, Infrared

splitting layer and the early exit threshold. Other studies, such as AgileNN [51] and DeepCOD [52], compress the intermediate features to be offloaded, thereby further reducing offloading latency. However, most existing works are designed for task-specific NNs that operate in a closed-set manner. Developing an edge-cloud collaborative system with open-set capability remains unexplored.

In summary, existing works either focus on optimizing the on-device inference for closed-set NNs or compressing FMs with dedicated designs that are static and not scalable for dynamic IoT applications. Although recent studies have explored efficient LLM inference on edge devices [34], [35], [36], they remain limited to text modalities and do not consider dynamic environments or multimodal sensing in embedded systems. We summarize recent work on efficient FMs in edge systems in Table I. Leveraging the open-set knowledge of FMs for embedded systems remains an unresolved issue.

III. BACKGROUND AND MOTIVATION

This section first introduces the background of FMs, then discusses the challenges of using FMs in embedded systems and studies the characteristics of FMs and closed-set models. Finally, we investigate the feasibility of using FMs as a knowledge base to customize edge models for open-set capability.

A. Multi-Modal Foundation Models

Foundation models (FMs) refer to a new class of large machine learning models that can extract valuable features to support diverse downstream tasks such as conversational systems [12] and zero-shot IoT sensing [15]. In general, FMs can be categorized into encoder-based FMs [11], [28] and multi-modal large language models (MLLMs) [10], [14].

Encoder-based FMs represented by CLIP and ImageBind [11], [28] learn the pairing of data of different modalities (e.g., RGB image and audio) and their corresponding textual description across the internet to achieve the capability of *open-set recognition*. In particular, encoder-based FMs adopt a transformer-based encoder to extract features from the raw data and convert them into *embedding*. FMs also use a text encoder to extract *text embedding* from the corresponding textual description of the raw data (e.g., images). The training of such multi-modal FMs often adopts contrastive learning to study the pairing between the data embedding and the corresponding text embedding to construct a *unified embedding space*. For any class

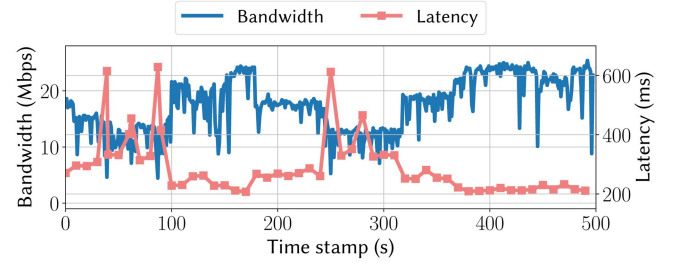


Fig. 2. An example showing the inference latency of cloud-centric FM solutions under dynamic network conditions.

described in a natural language manner, the text descriptions and the sensor data (such as images) are fed into the text encoder and image encoder of FMs to obtain respective embeddings. After computing the similarity score between the text embedding and other sensor data's embedding, FMs select the class with the highest score as the final prediction.

Although encoder-based FMs can achieve zero-shot IoT sensing, their architectures support only embedding extraction rather than open-ended language generation. Recent MLLMs, such as LLaVA [30] and Qwen-VL [31], further connect these encoder-based FMs to LLMs by mapping multi-modal embeddings into the embedding space of LLMs. This enables MLLMs to process multi-modal sensor data and generate open-ended language in an end-to-end manner. This capability enhances their generalization and practicality in embedded systems. However, the integration of additional heavyweight LLMs, which have significantly larger parameters and resource requirements compared to the multi-modal encoders, poses greater challenges for deployment on edge devices.

B. Cloud-Centric FM Services

Open-set capabilities are fundamental to embedded systems, yet most existing FM-serving systems employ cloud-centric solutions [53]. We first study the performance of cloud-centric FM serving on embedded systems. We establish an edge platform in an office room with an NVIDIA Jetson Nano to stream RGB images to a cloud server for object recognition. The cloud server employs ImageBind as the FM. As shown in Fig. 2, the system latency fluctuates significantly under varying network conditions, ranging from 200 ms to 630 ms. Since cloud-centric FM serving systems require streaming all user data to the cloud for inference, it can lead to significant system delays, especially under unstable and limited network conditions. This can greatly affect user experience, such as increasing the likelihood of collisions for home service robots.

C. Understanding FMs and SMs

We then study the performance of FMs and closed-set small models (SMs) using unseen test samples from two datasets. The setting follows real-world embedded systems, where SMs with closed-set capabilities are typically deployed on the edge. Specifically, we use the FLO102 dataset [54], which contains 102 flower classes, and a self-collected dataset with 40 common

TABLE II
THE PERFORMANCE OF SMS AND FMS WITH UNSEEN TEST SAMPLES ON FLO102 AND SELF-COL DATASETS, RESPECTIVELY

	Models	FLO102	Self-Col	Param	Latency
SMs	MobileNet	1.1%	2.6%	3.5M	36.8ms
	ResNet18	0.4%	3.4%	11.7M	30.5ms
FMs	ImageBind	78.4%	71.3%	1172M	N.A.
	CLIP-L/14	79.5%	77.1%	407.8M	N.A.
	LLaVA-1.6-34B	60.5%	82.5%	34B	N.A.

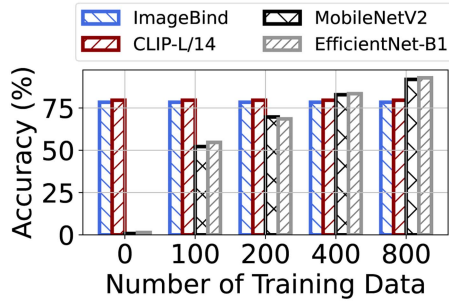


Fig. 3. The performance of SMs with different amounts of fine-tuned data. The parameters of FMs are frozen.

household items. We test two encoder-based FMs, CLIP [28] and ImageBind [11], as well as an MLLM, LLaVA [30]. We use MobileNetV2 [55] and ResNet18 [56] as SMs. We study how SMs and FMs perform on unseen samples, the customization of SMs to adapt to a dynamic set of classes, and measure their inference efficiency.

Open-set Capability: First, we divide the datasets into two subsets with non-overlapping classes to evaluate the open-set capabilities of SMs and FMs. As shown in Table II, FMs can reach a mean accuracy of up to 77% without any fine-tuning or calibration, while SMs average only 1.5%, comparable to random guessing. This significant performance gap indicates that FMs can effectively handle open-set scenarios, whereas SMs are largely ineffective for those unseen test samples.

Potential of Specialized SMs: Next, we measure the performance of the SMs by calibrating them with data from the new classes (i.e., the new types of objects added to the environment). Fig. 3 shows the performance of zero-shot FMs (i.e., without using any labeled data for calibration) and SMs customized with varying amounts of labeled data. Initially, SMs exhibit significantly lower performance compared to FMs when labeled data is limited. However, as the amount of labeled data for calibration increases, SMs can achieve comparable or even superior performance, demonstrating the potential of fully specialized SMs for specific tasks.

Inference Latency: Table II shows the model parameters and execution latency of SMs and FMs. The parameter size of FMs (LLaVA-1.6-34B) can be up to 9×10^3 times larger than that of SMs. The inference time of SMs on a Jetson Nano is less than 40ms, whereas FMs are hard to deploy due to memory constraints (N.A. in Table II). The high computational demands of FMs typically necessitate cloud deployment.

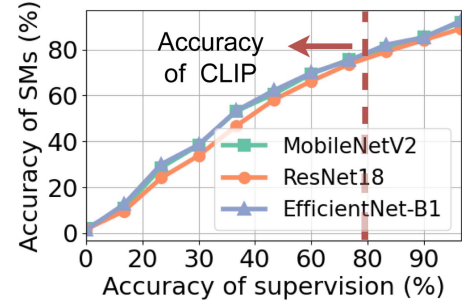


Fig. 4. Feasibility of leveraging FMs as a knowledge base for customizing small models. SM means small models.

In summary, FMs possess strong open-set capabilities but are too large to fit into embedded systems. On the other hand, efficient SMs can achieve better performance than FMs after being fully customized with labeled data from new classes. However, labeled data for customization is typically unavailable in practical embedded system applications. The above results motivate us to develop a collaborative system that fully leverages the generalization capabilities of FMs on the cloud and the efficiency of specialized SMs on the edge.

D. Customization With FMs

We further explore the feasibility of leveraging FMs to customize SMs by using the predictions of FMs as pseudo labels to supervise the customization of SMs. Specifically, we train SMs using different percentages of correct labels. Fig. 4 shows the performance of SMs when we set different accuracy of the pseudo labels. We control the correct percentage of the pseudo labels among 400 unseen test samples, which is the values on the x-axis. The vertical line shows the accuracy of CLIP-L/14. The result shows that the accuracy of the SM is 90% when all the pseudo labels are correct, while the accuracy of the SM drops to 80% when the pseudo labels have 80% accuracy. Meanwhile, CLIP can provide accuracy with 79.5% without any fine-tuning. This motivates us to use FMs as a rich and general knowledge base to provide high-quality supervision for the customization of SMs.

IV. APPLICATION AND SYSTEM OVERVIEW

DeepEdgeFM is a novel edge-cloud collaborative system with FMs that enables open-set capabilities in embedded systems. This section introduces the application scenarios and the overall system architecture of DeepEdgeFM.

A. Application Scenarios

DeepEdgeFM is designed to support a wide range of IoT applications that leverage FMs and MLLMs, where edge devices face computational or latency constraints. In particular, it enables edge-side open-set recognition *without any manual annotation* through continuous interaction between FMs and customized edge models. We demonstrate its applications through two examples. First, the robot performs semantic sensing, which

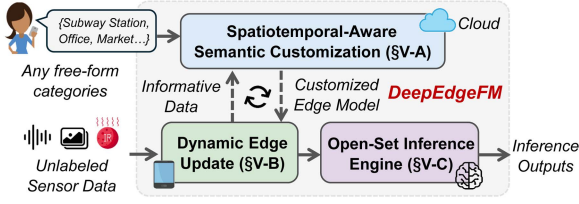


Fig. 5. System overview of DeepEdgeFM.

can be used in home services [9] or industrial logistics [57]. The key challenge is the diversity of personal items or goods, which can frequently change and encompass many different classes. However, existing systems require manually labeled data for newly added classes to adapt, significantly hindering their practical use [9]. The second example application is human activity recognition (HAR) on wearable devices like smartwatches, which can be used for disease monitoring and fitness tracking [2]. The main challenge is that current HAR systems require a model calibration phase, which involves collecting human-labeled data to adapt to users and their specific activities of interest. Additionally, the interested activities can change due to emerging health conditions or physicians' advice. Both example applications require recently emerged FMs with strong generation capabilities to provide open-set recognition on resource-limited embedded systems.

B. System Overview

Fig. 5 illustrates the system overview of DeepEdgeFM. Specifically, DeepEdgeFM supports open-set learning in IoT applications with categories articulated in any free-form natural language. DeepEdgeFM first employs a *spatiotemporal-aware semantic customization* approach (Section V-A) to adapt the edge model using FMs with heterogeneous architectures. It integrates spatial and temporal cues with FM-derived knowledge to continuously customize the edge model as the environment evolves. DeepEdgeFM also performs *dynamic edge update* (Section V-B), identifying informative unlabeled sensor data via confidence calibration and using a content-aware uploading strategy to reduce system overhead. DeepEdgeFM periodically update the customized edge model and text embedding pool to the edge to enable open-set recognition. During the inference phase, DeepEdgeFM's *inference engine* (Section V-C) considers both the uncertainty of the sensor data and network conditions to dynamically switch between the edge model and the FM, maintaining accuracy close to the original FM while significantly reducing cloud FM access.

V. DESIGN OF DEEPEGEFM

A. Spatiotemporal-Aware Semantic Customization

DeepEdgeFM employs a customized edge mode and dynamically queries cloud-side FM knowledge for customization. However, the continuously evolving environment and the heterogeneous architectures between FMs and closed-set models pose challenges for customization. To address these challenges,

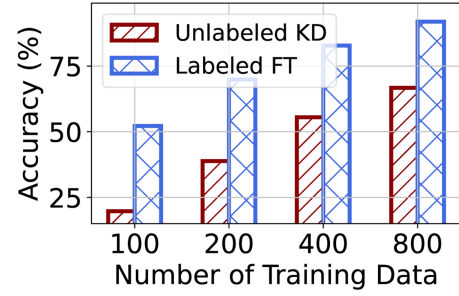


Fig. 6. Accuracy with labeled data and unlabeled data. FT denotes fine-tuning.

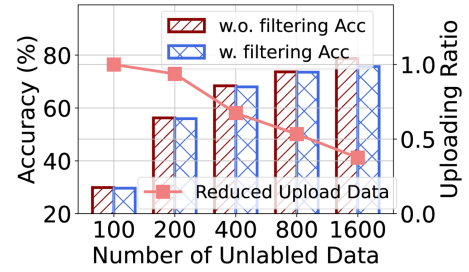


Fig. 7. Accuracy and uploading data ratio with or without content-aware data-uploading.

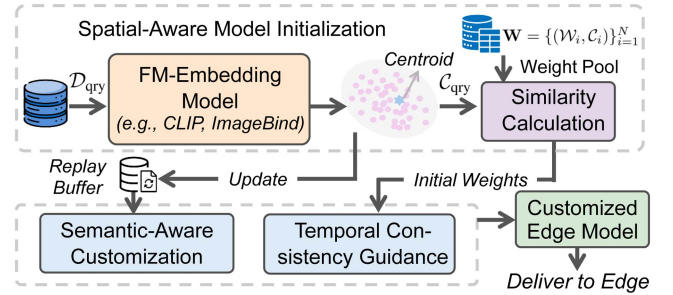


Fig. 8. Spatiotemporal-aware semantic customization.

we develop a spatiotemporal-aware semantic customization approach, shown in Fig. 8, which harnesses both the spatial and temporal cues to continually adapt domain-specific knowledge for the edge model in a label-free manner.

1) *Semantic-Aware Customization From FMs*: Existing work mainly focuses on distilling knowledge between homogeneous architectures [58]. However, most current FMs utilize Transformer architectures [11], while the limited computational resources of edge devices allow edge devices only support lightweight CNNs rather than heavyweight Transformers. This poses challenges in customizing lightweight CNNs using knowledge from heterogeneous FMs in a *label-free manner* while preserving open-set capability. To address this challenge, we propose a semantic-driven customization approach.

Heterogeneous Feature Projection: FMs encode the input images or spectrograms into a sequence of tokens and map them into a unified embedding space aligned with the text. The

difference in embedding space and heterogeneous model architectures between FMs and edge models makes the customization challenging. DeepEdgeFM employs lightweight CNNs as edge models. Lightweight CNNs [55] can be viewed as comprising a convolution-based feature extractor and a task-specific classifier. To this end, we discard the task-specific classifier of the original edge models and add a feature projection layer on top of the original feature extractor of edge models, which is defined as $\mathbf{v}_i = \psi(\mathcal{S}(\mathbf{x}_i))$, where $\mathcal{S}(\cdot)$ denotes the feature extractor of the customized edge model, $\psi(\cdot)$ is the feature projection layer. The architecture of the feature projection layer $\psi(\cdot)$ is a lightweight single-layer feed-forward network. It can ensure the features of the customized edge model have the same dimension as FM's unified embedding space, facilitating matching with text embeddings of FMs to enable open-set recognition.

Knowledge Query from the Foundation Model: Obtaining high-quality annotated sensor data is challenging in embedded systems. In such unlabeled scenarios, a direct approach to distilling task-specific knowledge from FMs is to use Mean Squared Error (MSE) loss to align the sensor data embeddings between the customized edge model and the foundation models. However, Fig. 6 shows that using unlabeled data and MSE loss for knowledge distillation leads to notable accuracy degradation compared to fine-tuning with labeled data.

To address this challenge, we propose to leverage FMs to further customize the *user-specific knowledge* to the edge models. Here we define the user-specific knowledge as the interested class set specified by users. Specifically, DeepEdgeFM initially pre-stores a *text embedding pool* that contains text embeddings from a wide range of frequently-used classes. The text embeddings in the pool are computed by the text encoder of FMs on the cloud. In practical use, DeepEdgeFM allows users to freely add their interested classes for respective applications. The text embeddings of these newly added classes are computed by FMs on the cloud and are added to the pool, which is then updated to the user device periodically (Section V-B3). Note that it does not mean requiring users to annotate each data, but only providing interested classes set. Take vision tasks as an example, suppose the text embedding pool is $\mathbf{T}$ and the visual encoder of FM is $\mathcal{T}_v(\cdot)$, FMs on the cloud first extract the visual embedding of the unlabeled data $\mathbf{x}_i$ as $\mathcal{T}_v(\mathbf{x}_i)$. Then, DeepEdgeFM will select a text embedding $\mathbf{t}'_i$ from the text embedding pool with the highest similarity with $\mathcal{T}_v(\mathbf{x}_i)$ as:

$$\mathbf{t}'_i = \operatorname{argmax}(\langle \mathcal{T}_v(\mathbf{x}_i), \mathbf{t}_k \rangle), \mathbf{t}_k \in \mathbf{T}, \quad (1)$$

where $\mathbf{t}'_i$ is the pseudo text embedding. We also assign $\mathbf{t}'_i$ a confidence score as $w_i = \langle \mathcal{T}_v(\mathbf{x}_i), \mathbf{t}'_i \rangle$, i.e. the cosine similarity between $\mathcal{T}_v(\mathbf{x}_i)$ and $\mathbf{t}'_i$. The pseudo text embedding and its confidence score will be used for semantic-driven distillation.

Free-Form Open-Set Learning: Although FMs like CLIP [28] and ImageBind [11] support open-set recognition, encoder-based FMs still rely on a predefined text-embedding pool, limiting their generalizability. To address this, we extend DeepEdgeFM to MLLMs, which support open-set recognition through freely generated natural-language outputs. However, MLLMs generate free-form text descriptions. Such heterogeneity in model architectures and output spaces poses challenges for

model customization. To address this, we employ a progressive customization approach. Specifically, we prompt the MLLM with “Please describe this sensor data in one word or phrase that best captures its category.” to generate a concise description of the current sensor data while remaining aligned with the closed-set label space, yielding pseudo-class labels. Next, we use the FM text encoder to embed these pseudo labels into pseudo text embeddings $\mathbf{t}'_i$ and compute their confidence scores w_i , as in (1).

Semantic-Aware Distillation Loss: To further customize the user-specific knowledge, we propose a semantic distillation loss. Here we take vision recognition as an example. Firstly, we adopt MSE loss to pull the features extracted by customized small models closer to the visual embedding of FMs, i.e. $\mathcal{L}_{vis} = \mathcal{H}_{MSE}(\mathcal{T}_v(\mathbf{x}_i), \mathbf{v}_i)$, where $\mathcal{T}_v(\mathbf{x}_i)$ and $\mathbf{v}_i$ are the visual embedding of FMs and customized small model, respectively; $\mathcal{H}_{MSE}$ is MSE function. Next, we adopt a bidirectional contrastive learning loss [28] to further pull the features extracted by customized small models closer to the pseudo text embedding of FMs. Given a mini-batch of bs paired data $(\mathbf{v}_i, \hat{\mathbf{t}}_i)$, where $\mathbf{v}_i$ is the embedding of the sensor data (e.g., image) extracted by the customized small model, and $\hat{\mathbf{t}}_i$ is the most similar text embedding in the text embedding pool of FMs, as in (1). We adopt the bidirectional contrastive learning loss which is defined as:

$$\mathcal{L}_i^{v \rightarrow t'} = -\log \frac{\exp \{ \langle \mathbf{v}_i, \hat{\mathbf{t}}_i \rangle / \tau \}}{\sum_{k=1}^{bs} \exp \{ \langle \mathbf{v}_i, \hat{\mathbf{t}}_k \rangle / \tau \}} \quad (2)$$

$$\mathcal{L}_i^{t' \rightarrow v} = -\log \frac{\exp \{ \langle \hat{\mathbf{t}}_i, \mathbf{v}_i \rangle / \tau \}}{\sum_{k=1}^{bs} \exp \{ \langle \hat{\mathbf{t}}_i, \mathbf{v}_k \rangle / \tau \}} \quad (3)$$

$$\mathcal{L}_{text} = \frac{1}{bs} \sum_{i=1}^{bs} w_i \{ \lambda \mathcal{L}_i^{v \rightarrow t'} + (1 - \lambda) \mathcal{L}_i^{t' \rightarrow v} \} \quad (4)$$

where w_i is the confidence score of the sample $\mathbf{x}_i$, which is obtained as mentioned before. λ and τ are the weight and temperature parameters, respectively. We test the two parameters with FLO102 dataset and choose $\lambda = 0.5$ and $\tau = 1$ that achieve the best recognition accuracy.

2) Spatial-Aware Model Initialization: Directly customizing the edge model from scratch or using checkpoints on public datasets is inefficient and can lead to long training times. In fact, cloud services for FMs, such as Microsoft Azure [18], typically serve multiple users with similar environments [59]. To further improve the accuracy and efficiency of customization, we integrate spatial cues into semantic-aware customization, helping determine the initial weights and architecture of the edge model. Next, we will first introduce the weight initialization, followed by architecture selection.

DeepEdgeFM leverages model weights from similar environments to improve the efficiency of edge model customization. Specifically, DeepEdgeFM maintains a server-side weight pool $\mathbf{W} = \{(\mathcal{W}_i, \mathcal{C}_i)\}_{i=1}^N$, where each model weight $\mathcal{W}_i$ is paired with an FM-derived centroid embedding $\mathcal{C}_i$. The embedding $\mathcal{C}_i$ is computed as the centroid of the FM embeddings extracted

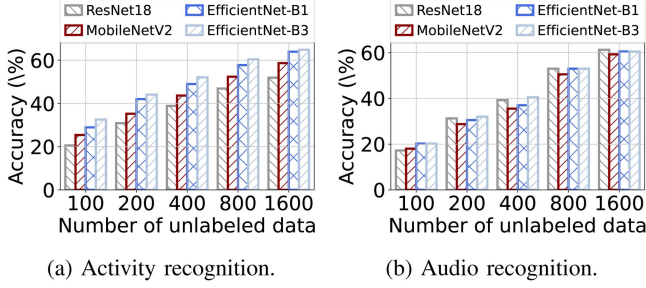


Fig. 9. Performance of customized edge models with different architectures on different modalities.

from its associated sensor data $\mathcal{D}_i$:

$$C_i = \text{Centroid}(\{\text{Enc}(\mathbf{d}) \mid \mathbf{d} \in \mathcal{D}_i\}), \quad (5)$$

where $\text{Enc}(\cdot)$ denotes the sensor encoder of the FMs. During the online stage, DeepEdgeFM collects a batch of sensor data denoted as $\mathcal{D}_{\text{qry}}$, computes its centroid embedding C_{qry} using (5), and retrieves the model weights that achieve the highest similarity to the query embedding:

$$(\mathcal{W}^*, \mathcal{C}^*) = \arg \max_{(\mathcal{W}_i, \mathcal{C}_i) \in \mathbf{W}} \text{sim}(C_{\text{qry}}, C_i). \quad (6)$$

The retrieved weight $\mathcal{W}^*$ with the highest similarity is selected as the initial edge model for the current user. We use the widely used cosine similarity in this work for its efficiency.

DeepEdgeFM also selects an appropriate model architecture based on system requirements and task profiles from edge devices. As shown in Fig. 9, different architectures exhibit varying accuracy–latency trade-offs across tasks and modalities [15], DeepEdgeFM maintains an architecture zoo containing classical edge models with pre-profiled accuracy, latency, and resource usage. During the online stage, DeepEdgeFM filters architectures by task and modality, then selects the one that maximizes recognition accuracy under system constraints.

3) *Temporal Consistency Guidance*: Real-world environments evolve continuously, causing data shifts that lead to significant accuracy degradation in semantic-aware customization due to the inherent trade-off between preserving historical knowledge and adapting to newly arrived data. Although prior work has explored diverse continual learning approaches [60], [61], [62], they remain limited in supporting open-set customization between heterogeneous FMs and lightweight edge models. To address this, we develop a temporal consistency guidance approach that integrates historical knowledge from the edge model with semantic customization, to inject new FM knowledge while maintaining temporal consistency.

Specifically, DeepEdgeFM first maintains a cluster-based replay buffer to store key historical sensor data. However, directly storing all user-uploaded data in the buffer and using every historical sample for each semantic-aware customization step can pose significant challenges in storage and customization efficiency. To address this, we develop a cluster-based buffer update strategy for efficient management and storage of data for customization. Specifically, we periodically cluster the buffered data $\mathcal{D}$ into k groups using K-means [63]. Since FM embeddings

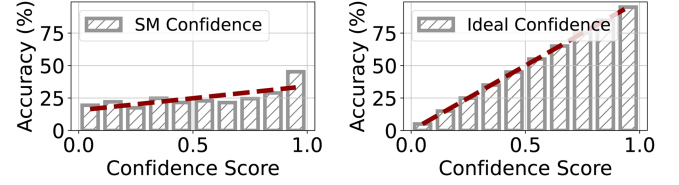


Fig. 10. Overconfidence phenomenon in the customized small model. The confidence score is high despite the low accuracy.

for all $\mathbf{d} \in \mathcal{D}$ are already computed during semantic-aware customization, K-means is performed directly in the FM embedding space. After clustering, only the samples whose embeddings are closest to their respective cluster centroids μ_j (i.e., $\arg \min_{\mathbf{d} \in \mathcal{D}_j} \|\text{Enc}(\mathbf{d}) - \mu_j\|$) are retained in the buffer for subsequent customization. The hyperparameter k controls the trade-off between customization accuracy and efficiency. We will discuss its impact in Section VI-D2.

Next, DeepEdgeFM performs history knowledge replay during the semantic customization. Although using filtered historical data during the customization process can somewhat alleviate the forgetting problem, there is still a performance gap compared to using all historical data for customization. To mitigate this issue, we utilize the previous knowledge of the history small model to improve customization efficiency. Specifically, we incorporate a constraint $\mathcal{H}_{MSE}(\mathbf{v}_i^{t-1}, \mathbf{v}_i^t)$ into the loss function to ensure that the small model does not forget historical data and knowledge during the customization process, where $\mathbf{v}_i^{t-1}$ and $\mathbf{v}_i^t$ are the visual embeddings from the small models at previous round of customization and this round, respectively. Therefore, the overall loss function of the small model customization in DeepEdgeFM is: $\mathcal{L} = \alpha \mathcal{L}_{FM} + (1 - \alpha) \mathcal{L}_{SM}$. The parameter α balances the trade-off between learning new knowledge and retaining previous knowledge during customization. It can be easily adjusted based on the specific application scenario. For example, in scenarios such as indoor robotic semantic sensing, where changes in data distribution are minimal, a smaller α value can be selected to prioritize retaining previous knowledge. We will discuss its impact in Section VI-D2.

B. Dynamic Edge Update

DeepEdgeFM continuously customize the edge model to dynamically adapt to the environment. However, streaming all data to the cloud can cause non-trivial transmission overhead and unreliability. Although prior works [64] have utilized softmax scores from conventional closed-set models to determine data uploading, the open-set features and overconfidence phenomenon of the edge model make existing solutions suboptimal and pose unique challenges. This section introduces the dynamic edge update in DeepEdgeFM, including confidence calibration and a content-aware data uploading strategy.

1) *Confidence Calibration*: The phenomenon of overconfidence refers to machine learning models exhibiting high confidence in their predictions, even though their predictions are incorrect [65]. In our experiments, we also observe that our semantic-driven customization algorithm can lead to the edge

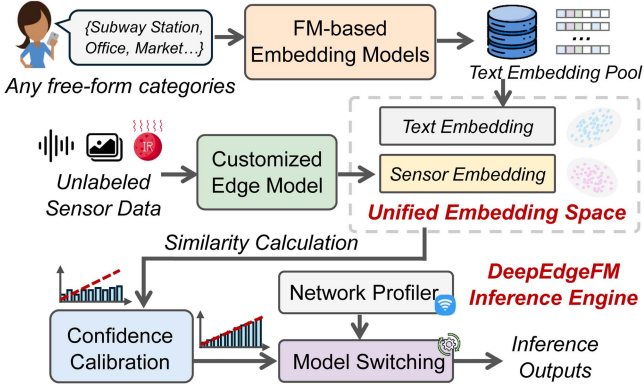


Fig. 11. Illustration of DeepEdgeFM's inference engine.

model exhibiting the overconfidence problem. Fig. 10 shows the relationship between the accuracy of customized small models and their predicted confidence scores on the UCF101 dataset [66], with the left and right figures showing the results from the small model and the ideal curve, respectively. When small models exhibit a confidence score above 0.5 for samples, their average classification accuracy only ranges from 20% to 40%, showing obvious overconfidence. The inaccurate confidence estimation of the small model on edge devices significantly impacts the edge-cloud collaboration system.

To address this challenge, we employ temperature scaling [65] to calibrate the confidence score in the predictions of the customized edge model. The confidence estimation of the edge model's predictions can be expressed as:

$$Conf(\mathbf{x}_i) = \frac{\exp\{\langle \mathbf{v}_i, \mathbf{t}_k \rangle / \tau c\}}{\sum_{\mathbf{t}_k \in \mathbf{T}} \exp\{\langle \mathbf{v}_i, \mathbf{t}_k \rangle / \tau c\}} \quad (7)$$

where $\langle \mathbf{v}_i, \mathbf{t}_k \rangle$ is the cosine similarity between sensor data embedding $\mathbf{v}_i$ (computed by the customized edge models) and each text embedding $\mathbf{t}_k$ in the text embedding pool $\mathbf{T}$. τc is the calibration parameter in temperature scaling. We set $\tau c = 5$ based on the observation that this configuration results in the best calibration performance of confidence, which can be seen in Section VI-D3 for more details.

2) *Content-Aware Data Uploading*: To reduce transmission overhead, DeepEdgeFM employs a content-aware data uploading strategy tailored for open-set models, utilizing the semantic similarity between sensor data embeddings and text embeddings as the criterion for uploading. In particular, we utilize the margin score [67] to quantify sensor data uncertainty, defined as $Unc(\mathbf{x}_i) = Conf_1(\mathbf{x}_i) - Conf_2(\mathbf{x}_i)$, where $Conf_1(\mathbf{x}_i)$ and $Conf_2(\mathbf{x}_i)$ denote the highest and second-highest similarities between the sensor data embeddings $\mathbf{v}_i$ and the text embeddings in $\mathbf{T}$, respectively. Sensor data with a low uncertainty score $Unc(\mathbf{x}_i) < U_{thre}$ will not be offloaded to the cloud for customization to save transmission cost, where U_{thre} is the threshold for data uploading. We empirically set this threshold to 0.99 as it exhibits the best performance. Fig. 7 shows that when the edge model is customized using 800 unlabeled sensor data in the current environment, the ratio of offloaded data can

be reduced by up to 50% with negligible accuracy degradation, significantly lowering data transmission overhead.

3) *Periodic Update on the Edge*: DeepEdgeFM periodically updates and delivers the customized small model from the cloud to the edge. The update interval is set to 200 seconds in the system, as existing studies [68] show that this interval achieves the best trade-off between accuracy and transmission overhead. Additionally, if the user's interested classes change, DeepEdgeFM will update and deliver the text embedding pool to the edge, enabling open-set recognition on edge devices.

C. DeepEdgeFM Inference Engine

Next, we introduce the inference engine of DeepEdgeFM, which dynamically switches between the customized edge model and the FMs based on the uncertainty of the sensor data and the real-time network conditions, as shown in the Fig. 11.

1) *Dynamic Model Switching*: DeepEdgeFM employs an edge-cloud hybrid inference strategy. For each sensor data $\mathbf{x}_i$, the overall predictions of DeepEdgeFM can be represented as follows:

$$P(\hat{y} | \mathbf{x}_i) = r(\mathbf{x}_i)P_{EM}(\hat{y} | \mathbf{x}_i) + (1 - r(\mathbf{x}_i))P_{FM}(\hat{y} | \mathbf{x}_i) \quad (8)$$

where $P_{EM}(\hat{y} | \mathbf{x}_i)$ and $P_{FM}(\hat{y} | \mathbf{x}_i)$ are the predictions of the customized edge model and FMs, respectively. Note that DeepEdgeFM is an edge-cloud collaborative system with open-set capabilities, thus both of $P_{EM}(\hat{y} | \mathbf{x}_i)$ and $P_{FM}(\hat{y} | \mathbf{x}_i)$ are calculated by the semantic similarities between sensor data embeddings and text embeddings: $P_{EM}(\hat{y} | \mathbf{x}_i) = \langle \mathbf{v}_i, \mathbf{t}_k \rangle$, $P_{FM}(\hat{y} | \mathbf{x}_i) = \langle \mathbf{T}_v(\mathbf{x}_i), \mathbf{t}_k \rangle$. Here, $\mathbf{v}_i$ and $\mathbf{T}_v(\mathbf{x}_i)$ are the sensor data embeddings from the edge model and FM, respectively, while $\mathbf{t}_k$ represents the text embeddings. Additionally, $r(\cdot)$ is the router model responsible for model switching, which can be defined as:

$$r(\mathbf{x}_i) = \mathbb{1}\{Unc(\mathbf{x}_i) \geq thre(t)\} \quad (9)$$

where $Unc(\mathbf{x}_i)$ represents the uncertainty of the sensor data. $thre(t)$ denotes the run-time model switching threshold, which is different from the threshold used for customization in Section V-B2. Additionally, it will be dynamically adapted to the network conditions (see Section V-C2 for details).

2) *Dynamic Network Adaptation*: The model switching threshold significantly impacts overall accuracy and system overhead. DeepEdgeFM employs a dynamic network adaptation strategy to adjust the switching threshold at run-time.

Specifically, DeepEdgeFM periodically collects sensor data from the environment and constructs a calibration set C , which is used for environment profiling. For each threshold $thre \in (0, 1)$, DeepEdgeFM measures the corresponding edge-side processing ratio $r(thre)$, corresponding average accuracy, and latency for the data in C , and saves this information in a threshold-searching table. During the run-time phase, the selection of $thre$ is determined by the priority of user demand. For accuracy-prioritized scenarios, DeepEdgeFM selects the smallest $thre$ that ensures no accuracy degradation. For the latency-prioritized cases, DeepEdgeFM determines the $thre$

based on:

$$\max_{thre \in (0,1)} thre \text{ s.t. } \hat{t}_{e2e}(thre) \leq L_{app} \quad (10)$$

where L_{app} is the latency constraints specified by the application or user. $\hat{t}_{e2e}(thre)$ is the end-to-end inference latency, which can be estimated by: $\hat{t}_{e2e}(thre) = r(thre) \cdot t_{edge} + (1 - r(thre)) \cdot (t_{trans} + t_{cloud})$, where t_{edge} and t_{cloud} are the latency on the edge and cloud side, respectively. The offloading latency t_{trans} can be estimated by $t_{trans} = \frac{DS}{B(t)}$, where DS is the size of the offloaded data, and $B(t)$ is the real-time network bandwidth. Note that network estimation has been widely studied in prior work [69]. Our approach is compatible with these existing solutions. Based on (10) and the equation for estimating $\hat{t}_{e2e}(thre)$, DeepEdgeFM can obtain the optimal edge processing ratio $r(thre)$. It then looks up the threshold-searching table to find the corresponding threshold. Utilizing these solutions, DeepEdgeFM can adapt to network conditions, dynamically adjusting the threshold at runtime.

D. System Implementation

1) *Edge-Cloud Implementation*: We deploy our system on a desktop server with 8 NVIDIA RTX A6000 GPUs and on two NVIDIA edge platforms, Jetson Nano and Jetson AGX Xavier. The network connection and data transmission parts in DeepEdgeFM are developed via TCP socket API. We use Traffic Control in Linux to simulate different network conditions and iPerf tool [70] to measure the network bandwidth at regular one-second intervals.

2) *Foundation Models*: We implement DeepEdgeFM on both encoder-based FMs and MLLMs. For encoder-based FMs, we utilize CLIP [28], ImageBind [11], and LanguageBind [29], which are among the most widely used multi-modal FMs. Specifically, we employ CLIP-L/14, the highest-performing model in the CLIP series. For MLLMs, we use LLaVA [30] and Qwen-VL [31] for evaluation. We adopt LLaVA-1.6-34b, the latest version in the LLaVA series. Additionally, since most FMs only support image and audio modalities, we leverage LanguageBind [29] to evaluate DeepEdgeFM's performance on depth and thermal modalities.

3) *Prompt Setting*: For CLIP, we utilize the text prompt template in [28]. For LanguageBind and ImageBind, we use the class names directly as the text prompt. For MLLMs, the prompt is set as: "Select the most likely category from the {Category List} that accurately describes the input data," enabling the MLLMs to perform the evaluation task.

4) *Baseline Approaches*: We compare DeepEdgeFM with two types of baselines, including the efficient on-device inference baselines and open-set recognition baselines.

Efficient On-device Inference Baselines: We implement several representative efficient on-device inference approaches on FMs to ensure a fair evaluation.

- *PersEPhonEE* [71], which is an edge-only NN acceleration approach based on early exit. we implement PersEPhonEE on the two FMs. There are two ways to implement the early-exit heads for encoder-based FMs, including a fully connected classifier and cosine distance classifier [72],

TABLE III
DETAILS OF DATASETS AND FMS USED IN EVALUATION. ①, ②, ③ ④, AND ⑤ DENOTE IMAGEBIND, CLIP, LLaVA, QWEN-VL, AND LANGUAGEBIND, RESPECTIVELY.

Dataset	Recognition Task	Modality	Samples	Classes	FMs
UCF101	Human Activity	RGB	11,331	101	①②③④
FLO102	Flower	RGB	8,189	102	①②③④
Self-Col	Indoor Scene	RGB	18,295	40	①②③④
Self-ColV2	Outdoor Scene	RGB	1,771	20	①
ESC50	Env. Sound	Audio	2,000	50	①
NYU-D	Indoor Scene	Depth	1,449	10	⑤
FLIR V2	Outdoor Object	Infrared	16,696	12	⑤

where the latter one is adopted as it performs better in experiments.

- *SPINN* [25], which is an edge-cloud collaboration approach integrating model splitting and early-exit techniques. We also adopt cosine distance classifier as the early-exit head.
- *Cloud-centric*, which is the most widely adopted solution for FM inference [19]. We deploy FMs on the server and offload all the samples to the server for inference.

We re-implemented PersEPhonEE and SPINN on the encoder-based FMs, CLIP and ImageBind. For MLLMs, we used a cloud-centric approach as the baseline.

Open-set Recognition Baselines: We also compare the performance of DeepEdgeFM with open-set recognition baselines.

- *Semantic-based Approaches*: DUS-VAE [37], ER-ZSAR [38], and VGGishZSL [73] are classic semantic-based baselines designed for HAR, audio recognition, and image recognition, respectively. They map the embeddings of sensor data to classes' text embeddings generated from Word2Vec [74] and BERT [75], thus achieving open-set recognition.
- *GAN-based approaches*: TF-VAEGAN [40] is a GAN-based open-set recognition approach, which uses a semantic decoder to synthesize features for unseen classes. As current GAN-based approaches mainly focus on vision tasks, they are not used for comparisons on tasks in other modalities.

VI. EVALUATION

A. Applications and Datasets

We evaluate DeepEdgeFM on five public datasets and a self-collected dataset with five FMs. Table III shows the details of the datasets and their corresponding FMs used for evaluation.

- *UCF101* [66]. This is a public video-based human activity recognition dataset. Following [28], we extract the middle frame of each activity video as input samples. It contains 11,331 images from 101 human activity categories. We use the same train-test split as in [28] for a fair evaluation.
- *FLO102* [54]. This is a public dataset that contains 8,189 images from 102 different categories of flowers. We use the same train-test split as in [28] for a fair evaluation.
- *ESC50* [76]. This is a public audio dataset that contains 2,000 audio clips from 50 classes of environmental sound.

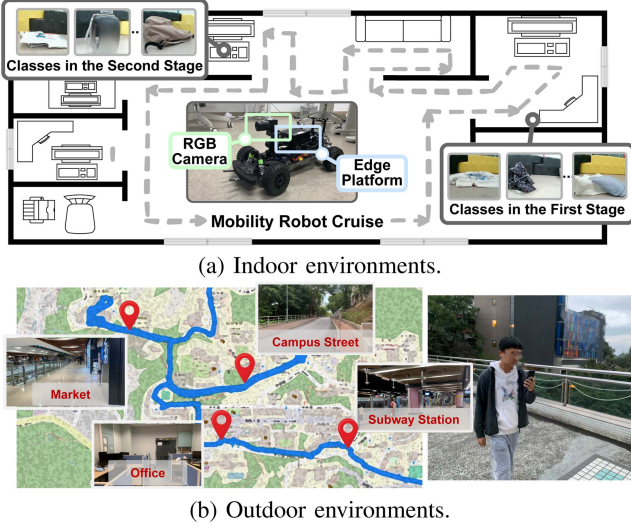


Fig. 12. Real-world testbed setups.

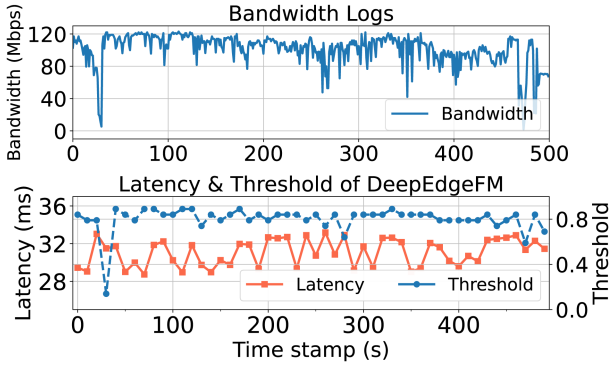


Fig. 13. Variation of network bandwidth, the threshold and inference latency of DeepEdgeFM.

Each audio clip is a 5-sec recording sampled at 44.1 kHz. We use the same train-test split as [73] for a fair evaluation.

- *NYU-D* [77]. This is a public indoor scene dataset with 1449 depth maps. It contains 10 categories of indoor scenes, such as bedrooms and offices. Following LanguageBind [29], we use 654 test samples in our experiments.
- *FLIR V2* [78]. This is a public infrared dataset that contains outdoor objects such as signs and bikes. Following LanguageBind [29], we use 16,696 thermal images across 12 categories for evaluation.
- *Indoor Self-collected Dataset (Self-Col)*: We collect a real-world indoor scene dataset, where objects in the environment are allowed to be added dynamically. As shown in Fig. 12(a), a robot equipped with a camera and edge platform moves randomly in the environment and takes RGB images continuously. We collect 18,295 RGB images in 40 classes. During the data collection, we place half of the classes first and place the other later. Current indoor scene datasets like SUN RGB-D [79] do not consider the scenario where users dynamically add objects into the environment, which is the main purpose of our self-collected dataset.

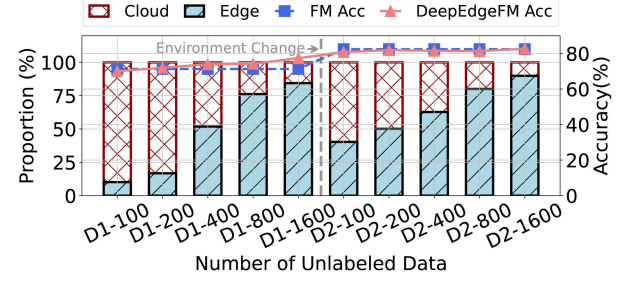


Fig. 14. DeepEdgeFM's performance when the environment changes in the end-to-end evaluation. X-axis shows the amount of collected data. D1 and D2 denote the two environments.

- *Outdoor Self-collected Dataset (Self-Col-V2)*: We also collected a real-world dataset in outdoor environments. A participant carried a smartphone during daily commuting and continuously captured environmental images. Fig. 12(b) shows the movement trajectory during commuting, covering various locations such as the dormitory, street, bus, supermarket, cafeteria, subway station, and office. The session lasted 28.3 minutes and produced a real-world dataset with 1,771 samples. Each frame was manually annotated into one of 20 scene categories following definitions in the Places365 dataset [80]. These categories include bus stops, supermarket interiors, and other common scenes.

B. End-to-End Evaluation

We first conduct end-to-end evaluations on two applications to validate the effectiveness of DeepEdgeFM in real-world scenarios. Fig. 12 illustrates the experimental setup, which includes a robot cruising through an indoor environment and a user carrying a smartphone for semantic sensing.

1) *Indoor Environments*: We first evaluate the performance of DeepEdgeFM in indoor environments. As shown in the upper part of Fig. 13, when the robot moves to different areas, the network bandwidth varies significantly, ranging from 2 Mbps to 123 Mbps. We set the mode to latency-prioritized and configure the latency constraint to 30 ms in the experiments, which satisfies most applications. The bottom part of Fig. 13 shows the threshold in DeepEdgeFM and the end-to-end inference latency. DeepEdgeFM sets a higher threshold at high network speeds to process more sensor data in the cloud (e.g., $t \in [100\text{ s}, 300\text{ s}]$), while it lowers the threshold at low network speeds to process more data locally (e.g., $t \in [20\text{ s}, 50\text{ s}]$). The results show that DeepEdgeFM can dynamically adjust its threshold for model switching, validating its effectiveness in adapting to varying network bandwidths.

We also evaluate the adaptability of DeepEdgeFM to environmental changes, including both the distribution and class changes in the sensor data. The objects from the first and second halves of the classes are sequentially placed into the environment. DeepEdgeFM operates in a completely unsupervised manner within this environment. Fig. 14 shows the processing ratio of sensor data on both the edge and cloud sides in DeepEdgeFM, as well as the overall accuracy. We also present the accuracy

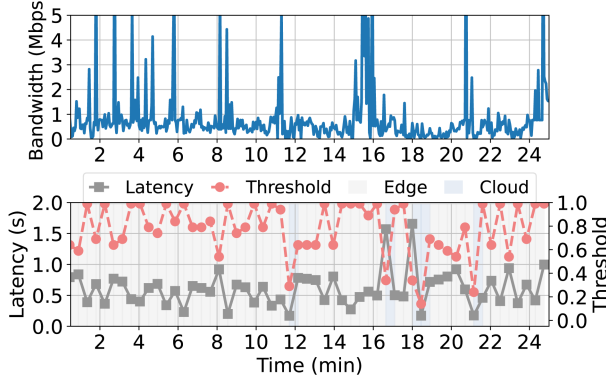


Fig. 15. End-to-end evaluation in outdoor environments. The bottom shows the inference latency, the threshold, and where each frame was processed on the edge or in the cloud.

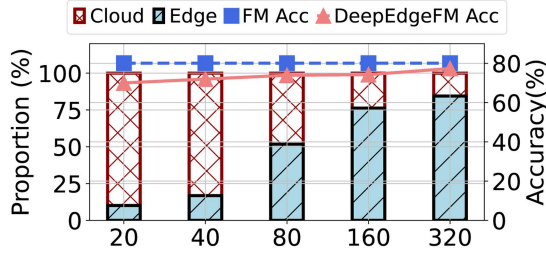


Fig. 16. Overall accuracy of DeepEdgeFM on the Self-ColV2 dataset and the edge-cloud processing rates. The x-axis is the number of unlabeled data used for edge model customization.

of the original FM as the upper performance bound. Fig. 14 shows that the edge-side processing ratio continuously increases as the amount of collected unlabeled sensor data grows, with negligible reduction in overall accuracy. At the gray dashed line, where the environment changes, DeepEdgeFM adaptively reduces the edge-side processing ratio from 84.4% to 40.2% to maintain overall accuracy. Overall, the results show that DeepEdgeFM can adapt to environmental changes at runtime by dynamically adjusting the edge-cloud processing ratio of the sensor data, ensuring that overall accuracy remains close to the original FM while significantly reducing system overhead.

2) *Outdoor Environments*: Next, we evaluate the performance of DeepEdgeFM in outdoor environments. Fig. 15 shows the real-world network bandwidth during the data collection session, where the smartphone was connected to the server via a 4G LTE network. Fig. 15 also shows DeepEdgeFM's end-to-end latency, the corresponding threshold variation over time, and whether each frame was processed on the edge or in the cloud. Results show that DeepEdgeFM processes most samples locally to provide fast responses, and only offloads samples to the cloud when they exceed the edge model's capability or when network bandwidth is sufficiently high. Fig. 16 shows the overall accuracy of DeepEdgeFM on the real-world dataset and the edge-cloud processing rates. The results indicate that DeepEdgeFM achieves accuracy comparable to the original FM while significantly reducing cloud queries. Specifically, up to

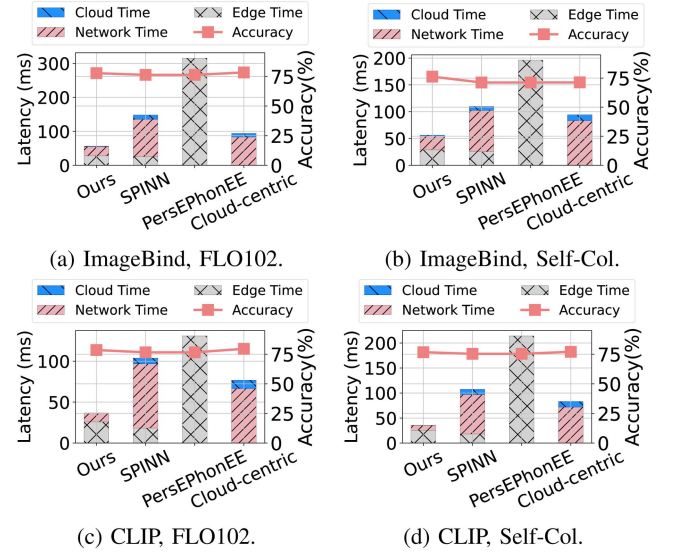


Fig. 17. Comparison of accuracy and inference latency.

TABLE IV
THE REDUCED END-TO-END LATENCY COMPARED WITH THE BEST BASELINES ON TWO EDGE PLATFORMS. N.A. MEANS NOT APPLICABLE SINCE CLIP SUPPORTS VISION DATA ONLY.

	FM	Device	FLO102	UCF101	Self-Col	ESC50
ImageBind	Xavier		1.7x	2.6x	1.7x	1.7x
	Nano		1.8x	2.4x	1.8x	1.3x
CLIP	Xavier		2.1x	2.2x	2.4x	N.A.
	Nano		2.5x	1.3x	3.2x	N.A.

84.4% of sensor data can be processed locally with negligible accuracy degradation.

C. Overall Performance of DeepEdgeFM

1) *Comparison With Baselines of Efficient On-Device Inference*: We first compare the performance of DeepEdgeFM with efficient NN inference baselines. The network bandwidth in the experiments is set to 55 Mbps, and this value is used throughout the paper unless otherwise specified. Fig. 17 and Table IV show that DeepEdgeFM achieves an end-to-end latency reduction of up to **1.3x** ~ **2.6x** for ImageBind and **1.3x** ~ **3.2x** for CLIP. Additionally, DeepEdgeFM achieves higher inference speedup compared to PersEPhoneEE and SPINN. This is because these two approaches require additional early-exit layers, which incur significant computational overhead in the context of FMs due to their high-dimensional embeddings and deep layers. Although SPINN partitions the inference between the edge and the cloud, offloading embeddings of FMs diminishes the benefits, as the embedding sizes are even larger than the raw sensor data. For example, the embedding of ImageBind is $257 \times 1 \times 1280$, while the raw image is $3 \times 224 \times 224$.

2) *Impact of Network Bandwidth*: Next, we evaluate the performance of DeepEdgeFM under various network conditions. We conduct experiments with bandwidths of 6 Mbps, 29 Mbps, and 55 Mbps, respectively. As illustrated in Fig. 18,

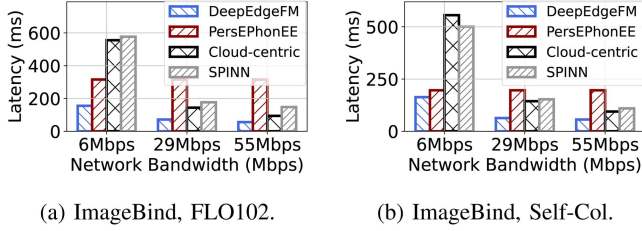


Fig. 18. Impact of network bandwidth on DeepEdgeFM.

TABLE V
THE REDUCED END-TO-END LATENCY OF DEEPEGEFM COMPARED TO CLOUD-CENTRIC METHODS WHEN APPLIED TO MLLMs

FM	Bandwidth	FLO102	UCF101	Self-Col	Avg.
LLaVA-1.6	50 Mbps	1.91x	1.30x	4.45x	2.55x
	10 Mbps	2.06x	1.33x	6.13x	3.17x
Qwen-VL	50 Mbps	1.47x	1.38x	3.37x	2.07x
	10 Mbps	1.80x	1.44x	4.53x	2.59x

DeepEdgeFM can achieve up to **3.5x** and **3.7x** end-to-end inference speedups on ImageBind compared to cloud-centric and SPINN approaches under 6 Mbps bandwidth. At higher network bandwidths, this gap narrows, with DeepEdgeFM still achieving up to **1.7x** and **2.4x** speedups compared to cloud-centric and SPINN under 55 Mbps bandwidth. Results show that DeepEdgeFM consistently achieves higher inference speedups under various network conditions, particularly under low bandwidth.

3) *Performance on MLLMs*: We further evaluate the performance of DeepEdgeFM on MLLMs. We use LLaVA-1.6-34b and Qwen-VL as the MLLMs for experiments. Since these MLLMs only support RGB modalities, the UCF101, FLO102, and Self-Col datasets are used for evaluation. We compare the end-to-end latency of DeepEdgeFM compared to cloud-centric solutions at different network bandwidths. The edge model in DeepEdgeFM was trained with 800 samples for each dataset. Table V shows that DeepEdgeFM can achieve average end-to-end latency reductions of up to 2.55x and 3.17x compared to the cloud-centric approach at network bandwidths of 50 Mbps and 10 Mbps, respectively, validating the effectiveness of DeepEdgeFM for MLLMs.

4) *Comparison With Open-Set Recognition Approaches on the Edge*: We also compare the open-set recognition capability of DeepEdgeFM with existing baselines. As illustrated in Table VI, DeepEdgeFM achieves accuracy gains of 20.8% and 21.7% over the top-performing baselines on the FLO102 and UCF101 datasets, respectively. Additionally, for the audio modality, DeepEdgeFM achieves a 34.3% improvement in accuracy compared to the best baseline. This is because these approaches lack generalizability due to their limited training data. Furthermore, DeepEdgeFM adopts edge-cloud collaborative inference strategy, significantly improving open-set recognition accuracy while maintaining comparable or even faster inference latency.

TABLE VI
OVERALL ACCURACY(%) AND LATENCY OF DEEPEGEFM COMPARED WITH OTHER OPEN-SET RECOGNITION APPROACHES

Dataset	Approach	Acc	Xavier	Nano
FLO102	TF-VAEGAN [40]	62.5%	54.2 ms	108.1 ms
	DUS-VAE [37]	62.1%	57.7 ms	110.1 ms
	DeepEdgeFM	83.3%	44.6 ms	62.2 ms
UCF101	TF-VAEGAN [40]	41.0%	53.9 ms	107.1 ms
	ER-ZSAR [38]	51.8%	87.9 ms	424.8 ms
	DeepEdgeFM	73.5%	42.7 ms	54.0 ms
ESC50	VGGishZSL [73]	33.0%	42.2 ms	217.2 ms
	DeepEdgeFM	67.3%	55.1 ms	67.5 ms

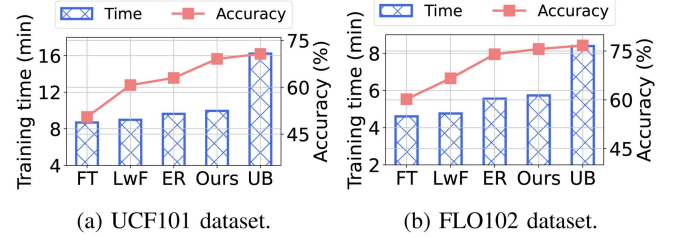


Fig. 19. Performance of customized edge models using spatiotemporal-aware semantic customization and baseline approaches. UB means the upper bound of accuracy.

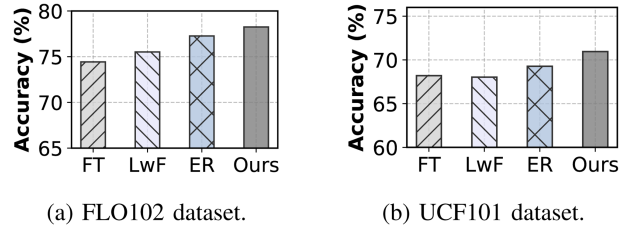


Fig. 20. Overall accuracy of edge-cloud inference with spatiotemporal-aware semantic customization and baselines.

D. Understanding DeepEdgeFM's Performance

1) *Effectiveness of Spatiotemporal-Aware Semantic Customization*: We test the accuracy of the customized edge model and customization time in the stream setting. Specifically, we split the dataset into two subsets according to the classes. Using the first subset, we first obtain the historical small model and the data buffer. Subsequently, we evaluate our streaming semantic customization approach on the second subset. We use fine-tuning (FT) [60], LwF [61], and ER [62] as baselines, where LwF and ER are representative approaches of regularization-based and replay-based methods in the lifelong learning domain, respectively. We also show the performance of using all the history data for customization, which can be seen as the upper bound (UB) [60] of accuracy.

Fig. 19 shows that our approach achieves up to 18.6%, 8.4%, and 6.1% accuracy gain compared with FT, LwF, and ER, respectively. In addition, compared with UB, our approach can reduce customization time by up to 1.6x with only 1.1% accuracy decrease. Fig. 20 shows that spatiotemporal-aware semantic customization can also improve the overall accuracy

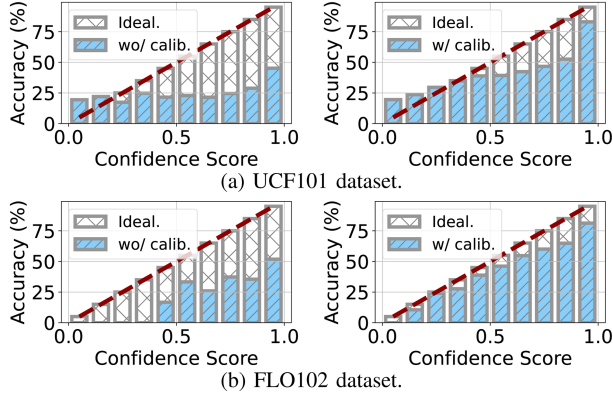


Fig. 21. Effectiveness of confidence calibration. Left and right figures show the accuracy of the customized edge models without and with calibration, respectively.

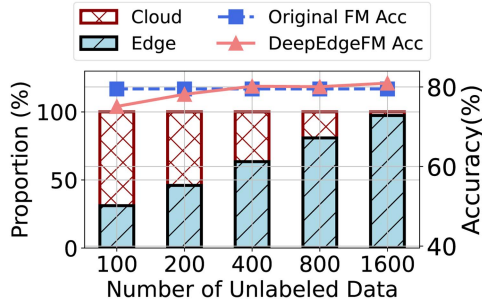


Fig. 22. Proportion of the data processed on the edge and the cloud, respectively.

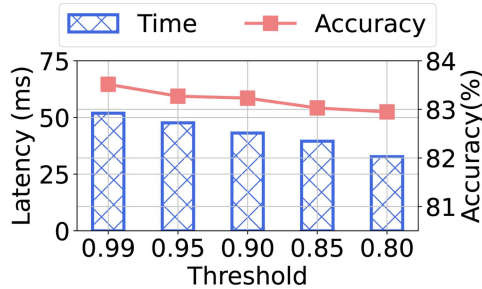


Fig. 23. Accuracy-latency trade-off caused by the confidence threshold.

of edge-cloud collaboration by up to 5.4% compared with the baselines, validating its effectiveness. Fig. 22 shows that more unlabeled data strengthens the customized edge model, enabling more data to be processed on-device.

2) *Impact of Hyperparameters:* We further study the impact of different hyperparameters in DeepEdgeFM. We first study the impact of weight λ in the semantic-aware distillation loss. Fig. 24 shows the experimental results on two datasets, UCF101 and FLO102. We trained the customized edge models using λ from 0.1 to 0.9. Results show that setting $\lambda = 0.5$ achieves the highest accuracy. Therefore, we set the hyperparameter $\lambda = 0.5$ in this work. In addition, we evaluate the hyperparameters in temporal consistency guidance, including the loss weight α and k . Fig. 25 shows that $\alpha = 0.01$ achieves the best accuracy, while Fig. 26 indicates that $k = 50$ provides the optimal

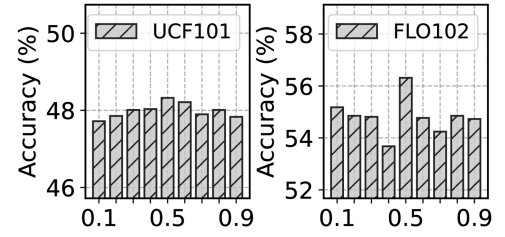


Fig. 24. Impact of weight λ in the semantic-aware distillation loss.

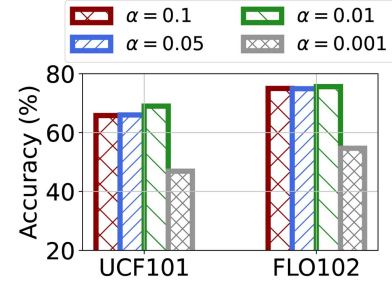


Fig. 25. Effect of weight α .

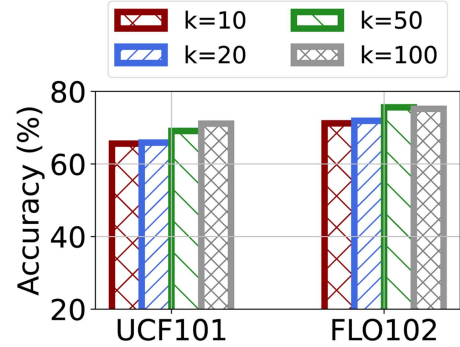


Fig. 26. Effect of k .

accuracy-latency trade-off. Therefore, we set $\alpha = 0.01$ and $k = 50$ in this study.

3) *Effectiveness of Confidence Calibration:* Next, we evaluate the effectiveness of confidence calibration. We conduct experiments on the UCF101 and FLO102 datasets and use ImageBind as FM. Fig. 21 demonstrates that the relationship between accuracy and confidence score more closely aligns with the ideal curve after calibration. The accuracy of the calibrated small model is 22.1% higher than that of the uncalibrated small model when the confidence score is 0.75. This accuracy gap can increase to 38.0% when the confidence score is 0.95. Results highlight the effectiveness of confidence calibration in DeepEdgeFM.

4) *Effectiveness of Spatial-Aware Model Initialization:* We further validate the effectiveness of the spatial-aware model initialization strategy in DeepEdgeFM. Specifically, we compare with multiple model matching baselines, including random model selection, training from scratch, and two methods that

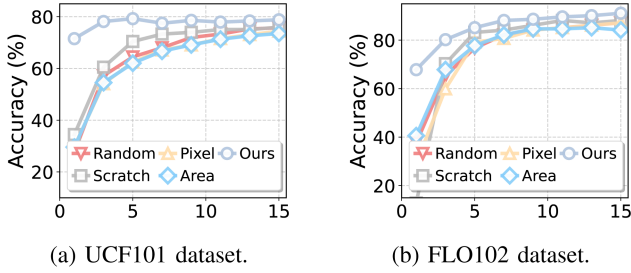


Fig. 27. Performance of spatial-aware model initialization. The x-axis denotes training epochs, and the y-axis shows the accuracy of the customized edge. Random and Scratch represent random model selection and training from scratch.

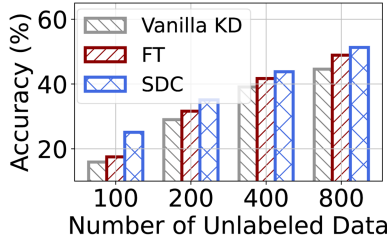


Fig. 28. Performance of semantic-aware customization.

select models using low-level image features [81], pixel similarity, and area similarity, denoted as *Pixel* and *Area*, respectively. We evaluate performance on both the FLO102 and UCF101 datasets. Fig. 27 illustrates the performance of the customized edge model using different initialization approaches during the training process. The results show that our approach significantly accelerates training and achieves up to 6.4% higher accuracy, thereby expediting customization and improving overall system efficiency.

5) *Effectiveness of Semantic-Driven Customization*: Finally, we evaluate the effectiveness of semantic-driven distillation for edge model customization. We compare our approach with two baselines: vanilla KD and fine-tuning (FT). Vanilla KD [82] utilizes the KL divergence to align the sensor data embeddings from the FM and the small model. FT uses hard pseudo labels from the FM for customization. Fig. 28 demonstrates that our semantic-driven customization (SDC) consistently outperforms the two baselines across different numbers of unlabeled data. This is because hard pseudo labels lack semantic relevance among classes [39], while vanilla KD fails to leverage the knowledge within text embeddings, resulting in inferior performance compared to our approach.

6) *Ablation Study*: We also conduct an ablation study to evaluate the effectiveness of the components in DeepEdgeFM. We show the testing results on the FLO102 dataset, since the results on the other datasets are similar. Fig. 30 shows the overall edge-cloud inference accuracy and the ratio of cloud queries when different components of DeepEdgeFM are removed. We evaluate semantic-driven customization (SE), spatial-aware model initialization (SP), temporal consistency guidance (TE), confidence calibration (CA), and architecture selection (AR).

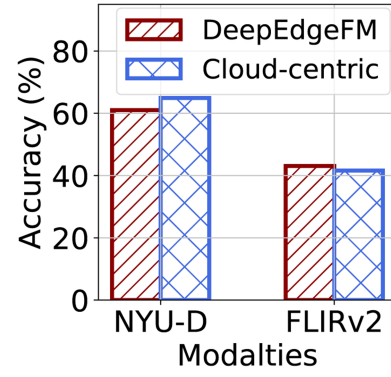


Fig. 29. Scalability to other modalities.

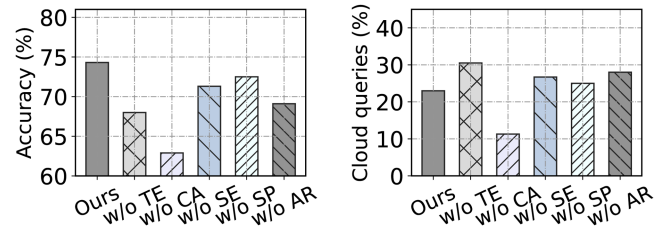


Fig. 30. Ablation study. The left and right figures show the overall accuracy and the ratio of cloud queries, respectively.

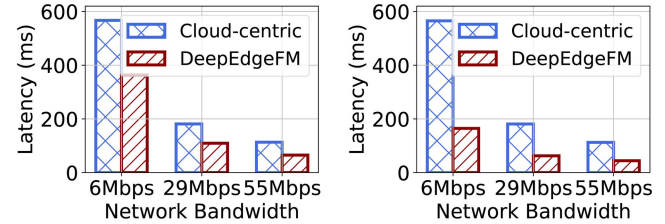


Fig. 31. Scalability to other modalities. The left and right figures are the inference latency of DeepEdgeFM and cloud-centric method on NYU-D and FLIR V2 datasets, respectively.

Results show that each component contributes substantial accuracy gains. We also observe that removing CA leads to severe overconfidence. Although only 11.3% of samples are processed locally, this results in a notable accuracy drop of 11.1%. These results further validate the effectiveness of the different components in DeepEdgeFM's design.

7) *Trade-Off Between Accuracy and Latency*: Fig. 23 illustrates the trade-off between accuracy and latency in DeepEdgeFM caused by the threshold. A lower threshold allows more data to be processed locally, speeding up end-to-end inference but potentially causing accuracy degradation. Conversely, a higher threshold can improve accuracy but requires more data to be offloaded to the cloud, which may not meet latency constraints in poor network conditions. DeepEdgeFM employs a dynamic network adaptation strategy to adjust the threshold. See Section VI-B1 for details.

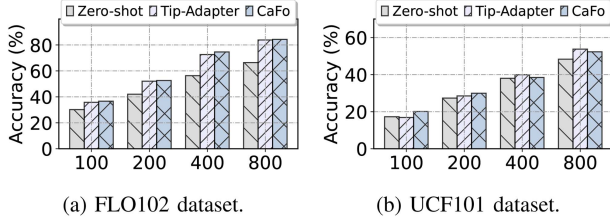


Fig. 32. Performance of DeepEdgeFM under unlabeled and labeled sensor data scenarios. The x-axis represents different amounts of sensor data, ranging from 100 to 800 samples. Zero-shot denotes using unlabeled sensor data.

E. System Scalability

This section evaluates the scalability of DeepEdgeFM across additional sensor modalities, FMs, and parameter-efficient fine-tuning (PEFT) approaches in supervised settings.

1) *Scalability to Other Modalities and FMs:* We then evaluate the performance of DeepEdgeFM on other modalities using the latest multi-modal FM, LanguageBind. We test the performance on the depth and infrared modalities (i.e., NYU-D and FLIRv2 datasets). Fig. 29 and Fig. 31 show the accuracy of DeepEdgeFM's customized edge models on these two modalities. We observe that DeepEdgeFM achieves relatively lower accuracy compared to its performance on RGB and audio modalities. This is because the underlying FM (i.e., ImageBind) still exhibits limited performance on these less common sensor modalities, due to the scarcity of training data compared with RGB and audio modalities. Fig. 29 shows the end-to-end latency of DeepEdgeFM on these two datasets under different network conditions. Under a 6 Mbps network condition, DeepEdgeFM reduces inference latency by **3.4x** and **1.6x** on the FLIR V2 and NYU-D datasets. Results validate the scalability of DeepEdgeFM to other modalities.

2) *Scalability to Supervised Scenarios:* Although our system is designed for unsupervised scenarios that leverage unlabeled sensor data for continuous edge-model customization, we also evaluate its scalability to supervised settings where partial data is annotated. We use two state-of-the-art parameter-efficient fine-tuning (PEFT) methods for FMs, Tip-Adapter [83] and CaFo [84], to first fine-tune CLIP with labeled data and then perform semantic-aware customization. Fig. 32 shows that using PEFT and labeled data further improves accuracy over using only unlabeled sensor data, validating the scalability of DeepEdgeFM to supervised scenarios.

F. System Overhead

We further evaluate DeepEdgeFM's system metrics. Specifically, we measured memory and energy usage on edge devices and FM query ratio and token consumption on the cloud side. We evaluated the performance on an NVIDIA Jetson Xavier using the FLO102 dataset, with the edge model customized using 200 unlabeled sensor samples. We compared DeepEdgeFM with the baseline PersEPhonEE [71], with both approaches using CLIP-ViT-L/14 as the FM. Fig. 33(a) shows that the edge-side memory and energy consumption of DeepEdgeFM are 22.4 MB and 1.4 J, achieving approximately 40x and 17x reductions,

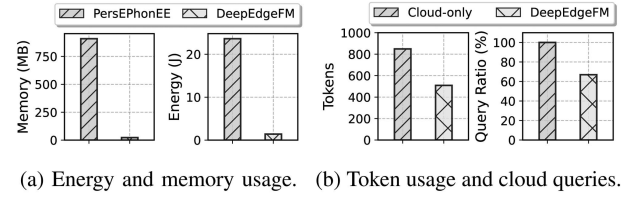


Fig. 33. Performance of DeepEdgeFM across system metrics.

respectively, compared to the baseline PersEPhonEE [71], which deploys the FM on the edge. Additionally, Fig. 33(b) shows that DeepEdgeFM can reduce the token consumption and cloud-side query ratio by 1.67x and 1.49x, respectively. Results validate the practicality of DeepEdgeFM in real-world use.

VII. DISCUSSION

Scalability to Complex Tasks: This study focuses on open-set recognition in embedded systems, where DeepEdgeFM maintains a customized small model for classification at the edge. With the increasing computational power of edge devices, smaller models such as 1B and 3B scale LLMs and MLLMs can now be deployed on these devices, enabling them to handle more complex tasks. Future work could further explore collaborative systems between LLMs and small language models for open-ended tasks like visual question answering and tool calling [85], [86], [87].

Scalability to Other Customization Strategies: We employ semantic-driven knowledge distillation to customize the edge models from FMs continuously in DeepEdgeFM. Other approaches like quantization [41] and MoE [50] can also be used to customize the small model. Further, DeepEdgeFM requires the cloud server to continuously deliver the updated edge model to the edge device. Existing techniques for partial model updating [88] can also be integrated into DeepEdgeFM to further reduce system communication overhead.

Optimal Edge Model Architecture: DeepEdgeFM maintains an architecture zoo for edge models, featuring a variety of classic model architectures. Although the zoo includes architectures searched by Neural Architecture Search techniques [89], they may not be the optimal student model architecture for the given FM. Future work could explore simultaneously customizing the model weights and the architecture for edge models during the customization process from the FM.

Customization with Labeled Sensor Data: DeepEdgeFM focuses on the unlabeled data scenario, which is considered a more challenging and practical setting for IoT applications. For scenarios with labeled or few-shot labeled data, these labeled data can first be used to train the FMs using a parameter-efficient fine-tuning (PEFT) approach [90]. The fine-tuned FM can then serve as an enhanced knowledge base for the edge model to query and customize, as in DeepEdgeFM.

VIII. CONCLUSION

In this paper, we introduce DeepEdgeFM, an efficient edge-cloud collaboration system with FMs that enables

open-set learning for IoT applications. DeepEdgeFM maintains the FMs on the cloud as a knowledge base and utilizes a lightweight model on the edge, which continuously customizes domain-specific knowledge from the FM. Meanwhile, DeepEdgeFM dynamically switches the model, taking into account both the uncertainty of sensor data and real-time network conditions. Comprehensive experiments show that DeepEdgeFM outperforms baselines with up to 3.2x lower latency and 34.3% higher accuracy.

ACKNOWLEDGMENT

We are grateful to the editors and anonymous reviewers for their insightful comments and suggestions.

REFERENCES

- [1] "Revenues from the AI-driven hardware market worldwide from 2018 to 2025," [Online]. Available: <https://www.statista.com/statistics/1003890/worldwide-artificial-intelligence-hardware-market-revenues/>
- [2] X. Ouyang et al., "Cosmo: Contrastive fusion learning with small data for multimodal human activity recognition," in *Proc. 28th Annu. Int. Conf. Mobile Comput. Netw.*, 2022, pp. 324–337.
- [3] B. Yang, L. Liu, W. Wu, M. Zhou, H. Liu, and X. Ning, "BrainZ-BP: A noninvasive cuff-less blood pressure estimation approach leveraging brain bio-impedance and electrocardiogram," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–13, Dec. 2023.
- [4] S. Shi et al., "Soar: Design and deployment of a smart roadside infrastructure system for autonomous driving," in *Proc. 30th Annu. Int. Conf. Mobile Comput. Netw.*, 2024, pp. 139–154.
- [5] K. Liu et al., "Tasking heterogeneous sensor systems with LLMs," in *Proc. 22nd ACM Conf. Embedded Networked Sensor Syst.*, 2024, pp. 901–902.
- [6] B. Yang, W. Wu, Y. Liu, and H. Liu, "A novel sleep stage contextual refinement algorithm leveraging conditional random fields," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–13, Feb. 2022.
- [7] A. Akbari and R. Jafari, "Transferring activity recognition models for new wearable sensors with deep generative domain adaptation," in *Proc. 18th Int. Conf. Inf. Process. Sensor Netw.*, 2019, pp. 85–96.
- [8] T. Gong, Y. Kim, J. Shin, and S.-J. Lee, "Metasense: Few-shot adaptation to untrained conditions in deep mobile sensing," in *Proc. 17th Conf. Embedded Networked Sensor Syst.*, 2019, pp. 110–123.
- [9] I. Kishida, H. Chen, M. Baba, J. Jin, A. Amma, and H. Nakayama, "Object recognition with continual open set domain adaptation for home robot," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2021, pp. 1517–1526.
- [10] J. Achiam et al., "Gpt-4 Technical Report," 2023, *arXiv:2303.08774*.
- [11] R. Girdhar et al., "ImageBind: One embedding space to bind them all," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 15180–15190.
- [12] B. Yang et al., "DrHouse: An LLM-empowered diagnostic reasoning system through harnessing outcomes from sensor data and expert knowledge," *Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.*, vol. 8, no. 4, pp. 1–29, 2024.
- [13] B. Yang et al., "SocialMind: LLM-based proactive AR social assistive system with human-like perception for in-situ live interactions," *Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.*, vol. 9, no. 1, pp. 1–30, 2025.
- [14] B. Yang, L. He, K. Liu, and Z. Yan, "VIAssist: Adapting multi-modal large language models for users with visual impairments," in *Proc. 2024 IEEE Int. Workshop Found. Models Cyber-Phys. Syst. Internet Things (FMSys)*, 2024, pp. 32–37.
- [15] B. Yang et al., "EdgeFM: Leveraging foundation model for open-set learning on the edge," in *Proc. 21st ACM Conf. Embedded Networked Sensor Syst.*, 2023, pp. 111–124.
- [16] R. Bommasani et al., "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.
- [17] X. Li, X. Jiang, X. Meng, A. Sun, and Y. Wang, "FreeLM: Fine-tuning-free language model," 2023, *arXiv:2305.01616*.
- [18] "Azure openAI service," 2024. [Online]. Available: <https://azure.microsoft.com/en-us/products/ai-services/openai-service>
- [19] D. Shah et al., "LM-NAV: Robotic navigation with large pre-trained models of language, vision, and action," in *Conf. Robot Learn.*, 2023, pp. 492–504.
- [20] M. Ahn et al., "Do as i can and not as i say: Grounding language in robotic affordances," 2022, *arXiv:2204.01691*.
- [21] X. Sun, P. Zhang, P. Zhang, H. Shah, K. Saenko, and X. Xia, "DIME-FM: Distilling multimodal and efficient foundation models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2023, pp. 15521–15533.
- [22] Y. Wei et al., "Contrastive learning rivals masked image modeling in fine-tuning via feature distillation," 2022, *arXiv:2205.14141*.
- [23] C. Zhang et al., "Faster segment anything: Towards lightweight sam for mobile applications," 2023, *arXiv:2306.14289*.
- [24] Y. Kang et al., "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 615–629, Apr. 2017.
- [25] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "SPINN: Synergistic progressive inference of neural networks over device and cloud," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 1–15.
- [26] Z. Zhao, K. Wang, N. Ling, and G. Xing, "EdgeML: An automl framework for real-time deep learning on the edge," in *Proc. Int. Conf. Internet Things Des. Implementation*, 2021, pp. 133–144.
- [27] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30.
- [28] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.
- [29] B. Zhu et al., "LanguageBind: Extending video-language pretraining to N-modality by language-based semantic alignment," in *Proc. 12th Int. Conf. Learn. Representations*, 2024.
- [30] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 36, pp. 34892–34916.
- [31] J. Bai et al., "Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond," 2023, *arXiv:2308.12966*.
- [32] D. Xue, X. Fan, T. Chen, G. Lan, and Q. Song, "Leveraging foundation models for zero-shot iot sensing," in *Proc. 27th Eur. Conf. Artif. Intell.*, 2024, pp. 3805–3812.
- [33] Q. Zhang, R. Han, C. H. Liu, G. Wang, S. Guo, and L. Y. Chen, "EdgeTA: Neuron-grained scaling of foundation models in edge-side retraining," *IEEE Trans. Mobile Comput.*, vol. 24, no. 4, pp. 2690–2707, Apr. 2024.
- [34] R. Yi, L. Guo, S. Wei, A. Zhou, S. Wang, and M. Xu, "EdgeMoE: Empowering sparse large language models on mobile devices," *IEEE Trans. Mobile Comput.*, vol. 24, no. 8, pp. 7059–7073, Aug. 2025.
- [35] M. Zhang, X. Shen, J. Cao, Z. Cui, and S. Jiang, "Edgeshard: Efficient LLM inference via collaborative edge computing," *IEEE Internet Things J.*, vol. 12, no. 10, pp. 13119–13131, May 2025.
- [36] H. Zhan et al., "PICE: A semantic-driven progressive inference system for LLM serving in cloud-edge networks," 2025, *arXiv:2501.09367*.
- [37] H. Su, J. Li, Z. Chen, L. Zhu, and K. Lu, "Distinguishing unseen from seen for generalized zero-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 7885–7894.
- [38] S. Chen and D. Huang, "Elaborative rehearsal for zero-shot action recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 13638–13647.
- [39] M. T. Islam and S. Nirjon, "Soundsemantics: Exploiting semantic knowledge in text for embedded acoustic event classification," in *Proc. 18th Int. Conf. Inf. Process. Sensor Netw.*, 2019, pp. 217–228.
- [40] S. Narayan, A. Gupta, F. S. Khan, C. G. Snoek, and L. Shao, "Latent embedding feedback and discriminative features for zero-shot classification," in *Proc. Eur. Conf. Comput. Vis.—ECCV 2020*, 2020, pp. 479–495.
- [41] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4820–4828.
- [42] Q. Cao, P. Khanna, N. D. Lane, and A. Balasubramanian, "Mobivqa: Efficient on-device visual question answering," *Proc. the ACM Interactive, Mobile, Wearable Ubiquitous Technol.*, vol. 6, no. 2, pp. 1–23, 2022.
- [43] M. Yuan, L. Zhang, F. He, X. Tong, and X.-Y. Li, "Infi: End-to-end learnable input filter for resource-efficient mobile-centric inference," in *Proc. 28th Annu. Int. Conf. Mobile Comput. Netw.*, 2022, pp. 228–241.
- [44] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu, "Deepcache: Principled cache for mobile deep vision," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 129–144.
- [45] W. Niu et al., "Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2020, pp. 907–922.
- [46] W. Kwon et al., "Efficient memory management for large language model serving with pagedattention," in *Proc. ACM SIGOPS 29th Symp. Operating Syst. Princ.*, 2023, pp. 611–626.

- [47] Y. Wang, K. Chen, H. Tan, and K. Guo, "Tabi: An efficient multi-level inference system for large language models," in *Proc. Eighteenth Eur. Conf. Comput. Syst.*, 2023, pp. 233–248.
- [48] H. Xia, T. Ge, P. Wang, S.-Q. Chen, F. Wei, and Z. Sui, "Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation," in *Proc. Findings Assoc. Comput. Linguistics: EMNLP 2023*, 2023, pp. 3909–3925.
- [49] M. Elhoushi et al., "Layer skip: Enabling early exit inference and self-speculative decoding," in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics*, (Volume 1: Long Papers), 2024, pp. 12622–12642.
- [50] D. Lepikhin et al., "Gshard: Scaling giant models with conditional computation and automatic sharding," in *Proc. Int. Conf. Learn. Representations*, 2021.
- [51] K. Huang and W. Gao, "Real-time neural network inference on extremely weak devices: Agile offloading with explainable AI," in *Proc. 28th Annu. Int. Conf. Mobile Comput. Netw.*, 2022, pp. 200–213.
- [52] S. Yao et al., "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency," in *Proc. 18th Conf. Embedded Networked Sensor Syst.*, 2020, pp. 476–488.
- [53] G. Cloud and V. AI, 2021. [Online]. Available: <https://cloud.google.com/vertex-ai>
- [54] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 2008 Sixth Indian Conf. Comput. Vis., Graph. Image Process.*, 2008, pp. 722–729.
- [55] A. G. Howard et al., Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
- [56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [57] S. Vallachira, M. Orkisz, M. Norrlöf, and S. Butail, "Data-driven gearbox failure detection in industrial robots," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 193–201, Jan. 2020.
- [58] X. Jiao et al., "TinyBERT: Distilling BERT for natural language understanding," in *Proc. Findings Assoc. Comput. Linguistics: EMNLP 2020*, Nov. 2020, pp. 4163–4174.
- [59] M. Khani et al., "Recl: Responsive resource-efficient continuous learning for video analytics," in *Proc. 20th USENIX Symp. Networked Syst. Des. Implementation*, 2023, pp. 917–932.
- [60] M. Mirtaheri, M. Rostami, and A. Galstyan, "History repeats: Overcoming catastrophic forgetting for event-centric temporal knowledge graph completion," in *Proc. Findings Assoc. Comput. Linguistics: ACL 2023*, 2023, pp. 7740–7755.
- [61] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
- [62] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32.
- [63] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, Jun. 2010.
- [64] M. Li, Y. Li, Y. Tian, L. Jiang, and Q. Xu, "AppealNet: An efficient and highly-accurate edge/cloud collaborative architecture for DNN inference," in *Proc. 2021 58th ACM/IEEE Des. Automat. Conf.*, 2021, pp. 409–414.
- [65] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Int. Conf. Mach. Learn.*, 2017, pp. 1321–1330.
- [66] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.
- [67] E. Park et al., "Big/little deep neural network for ultra low power inference," in *Proc. 2015 Int. Conf. Hardware/Softw. Codesign System Synth.*, 2015, pp. 124–132.
- [68] R. Bhardwaj et al., "Ekya: Continuous learning of video analytics models on edge compute servers," in *19th USENIX Symp. Networked Syst. Des. Implementation*, 2022, pp. 119–135.
- [69] M. Kim and B. Noble, "Mobile network estimation," in *Proc. 7th Annu. Int. Conf. Mobile Comput. Netw.*, 2001, pp. 298–309.
- [70] D. Jon, E. Seth, B. A. Mah, P. Jeff, and P. Kaustubh, "iPERF," 2014. [Online]. Available: <https://software.es.net/iperf/>
- [71] I. Leontiadis, S. Laskaridis, S. I. Venieris, and N. D. Lane, "It's always personal: Using early exits for efficient on-device cnn personalisation," in *Proc. 22nd Int. Workshop Mobile Comput. Syst. Appl.*, 2021, pp. 15–21.
- [72] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," 2019, *arXiv:1904.04232*.
- [73] H. Xie and T. Virtanen, "Zero-shot audio classification via semantic embeddings," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 1233–1242, Mar. 2021.
- [74] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, vol. 26, pp. 3111–3119.
- [75] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. 2019 Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., Vol. 1 (Long Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.
- [76] K. J. Piczak, "ESC: Dataset for environmental sound classification," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 1015–1018.
- [77] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Comput. Vis.—ECCV 2012*, 2012, pp. 746–760.
- [78] FLIR Teledyne, "Free teledyne flir thermal dataset for algorithm training," 2015. Accessed: Sep. 16, 2023. [Online]. Available: <https://www.flir.com/oem/adas/adas-dataset-form/>,
- [79] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 567–576.
- [80] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, Jun. 2018.
- [81] Y. Li et al., "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proc. Annu. Conf. ACM Special Int. Group Data Commun. Appl., technol., Architectures, Protoc. Comput. Commun.*, 2020, pp. 359–376.
- [82] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learn. Representation Learn. Workshop*, 2015.
- [83] R. Zhang et al., "Tip-adaptor: Training-free adaption of CLIP for few-shot classification," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 493–510.
- [84] R. Zhang et al., "Prompt, generate, then cache: Cascade of foundation models makes strong few-shot learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 15211–15222.
- [85] B. Yang et al., "ContextAgent: Context-aware proactive LLM agents with open-world sensory perceptions," in *Proc. 39th Annu. Conf. Neural Inf. Process. Syst.*, 2025.
- [86] B. Yang et al., "ProAgent: Harnessing on-demand sensory contexts for proactive LLM agent systems," 2025, *arXiv:2512.06721*.
- [87] K. Liu et al., "Tasksense: A translation-like approach for tasking heterogeneous sensor systems with llms," in *Proc. 23rd ACM Conf. Embedded Networked Sensor Syst.*, 2025, pp. 213–225.
- [88] Z. Qu, C. Liu, and L. Thiele, "Deep partial updating: Towards communication efficient updating for on-device inference," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 137–153.
- [89] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.
- [90] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in *Int. Conf. Learn. Representations*, 2021.