Abstract The rise of on-device deep learning (DL) in resource-limited mobile and embedded devices has stimulated various applications. However, existing on-device DL mostly relies on *predefined* processing patterns for *reacting* to *given* input data, resulting in accuracy and resource efficiency bottlenecks. In this work, we advocate a paradigm shift towards Swarm DL, inspired by the collective intelligence observed in swarms, where individual *proactive* actions drive superior global performance. Harnessing the potential of swarms formed by physically adjacent mobile and embedded devices in IoT scenarios, we introduce DeepSwarm, a closed-loop system framework aiming to push the performance boundaries of on-device DL. In particular, DeepSwarm incorporates proactive data acquisition and processing with *bi-directional optimization* to minimize redundancy and enhance resource efficiency. It promptly addresses data limitations and redundancy through DL feedback, capitalizing on the complementary and asynchronous nature of data for scalable processing. Finally, we discuss research challenges and opportunities for enhancing DeepSwarm performance and showcase two initial instances.

Keywords On-device Deep Learning, Swarm Intelligence, Swarm Deep Learning

1 Introduction

On-device deep learning (DL) on mobile and embedded IoT devices has stimulated various applications [1]. Examples include health monitoring on smartphones and watches [2], image recognition on robotics [3], and object classification on drone swarms [4]. Executing advanced DL models on-device can process local data efficiently, preserving privacy, enhancing responsiveness, and conserving network bandwidth. However, existing on-device DL mostly relies on *predefined* processing patterns for *reacting* to given input data, resulting in accuracy and resource efficiency bottlenecks. For example, the redundancy or absence of input data poses challenges to DL-based data processing, thereby limiting overall performance. It is difficult to provide feedback on the data processing performance during the data acquisition stage, as data processing typically occurs after data acquisition.

Harnessing the potential of swarms formed by physically adjacent mobile and embedded devices in IoT scenarios, we advocate a paradigm shift towards *Swarm DL* by drawing inspiration from *swarm intelligence* [5]. This shift entails moving from *reactive* on-device DL, which responds to given input data, to *proactive* systems known as swarm deep learning (swarm DL). Conceptually, *swarm intelligence* involves the collective intelligent behavior of multiple agents, each acting *proactively* based on simple patterns in a self-organized, self-adaptive, self-evolved manner, leading to enhanced global performance. Building upon on-device DL as the foundation, Swarm DL ***proactively*** scales data acquisition and processing and provides ***bi-directional optimization feedback*** for them, forming a closed loop. This paradigm aims to *maximize implicit complementarity* and *minimize redundancy* in both data acquisition and processing within the swarm, fostering a more efficient and scalable IoT system. Specifically, Swarm DL realizes swarm intelligence by leveraging IoT devices equipped with advanced sensors and DL computing capabilities as ***proactive agents***, embodying following visions:

- *Proactive Data Acquisition for DL*: Different from the traditional paradigm where sensor data acquisition is **completed** before data processing **fixedly**, Swarm DL enables each IoT device to **leverage runtime feedback from DL-based data processing to address the absence and redundancy of data during data acquisition as soon as possible**. Each device operates proactively and locally, leveraging the local proxy of global data distribution and limited resources, to enhance collaborative performance in a self-organized manner and facilitate more accurate and efficient data acquisition.
- *Proactive Data Processing by DL*: To capitalize on the complementary and asynchronism, *i.e.*, system issue, of acquired data, Swarm DL ***proactively* scales up/down** on-device DL model sub-structures and underlying system configurations. This *local* adaptation optimizes *global* performance (*e.g.*, accuracy, latency, energy cost, memory usage), with awareness of dynamic resource availability over the swarm. Also, unlike solely executing on-device DL, *proactive* swarm agents possess **greater adaptability to non-stationary mobile environments** with data drifts via active learning.

Despite extensive research on on-device DL inference [6] and adaptation [7], extending them to incorporate bio-direction optimization of data acquisition and processing presents challenges. Two complexities reside:

- *From independent optimization of data acquisition & processing to bi-directional optimization.* It requires online profiling of data importance in modality and samples before exactly executing data processing on each device. It involves incorporating data dynamics among the swarm. Specifically, existing on-device data importance assessment methods only evaluate *explicit im-*

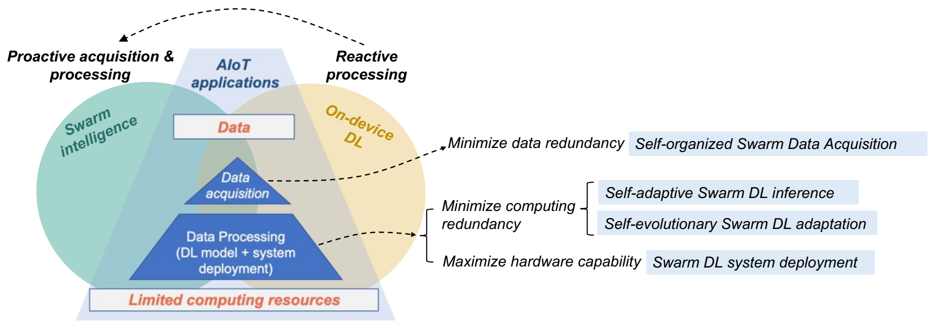

Proactive acquisition & processing
AIoT applications
Reactive processing
Swarm intelligence
On-device DL
Data
Data acquisition
Data Processing (DL model + system deployment)
Limited computing resources
Minimize data redundancy
Self-organized Swarm Data Acquisition
Minimize computing redundancy
Self-adaptive Swarm DL inference
Self-evolutionary Swarm DL adaptation
Maximize hardware capability
Swarm DL system deployment

Fig. 1: DeepSwarm integrates swarm intelligence into the on-device deep learning (DL) to enable *proactive* data acquisition and processing at resource-scarce agents while satisfying global application demands.

portance through distribution or entropy comparisons. They cannot handle *implicit complementarity*, influenced by distributed and dynamic factors like data asynchronism, cross-modal mutual information transferability, and associated latency costs for fusion

- *From reactive data processing to proactive data acquisition and processing*. Despite prior efforts in data-reactive on-device DL algorithm-system codesign, the optimization scope for proactive swarm devices needs refinement [38]. Decentralized constraints, such as processing from partial information and distributed resources, exacerbate optimization difficulties in swarm DL. Additionally, collaboration among individual IoT devices introduces new challenges for data acquisition association and computation optimization in swarm DL. In a distributed setting, the optimal matching of data acquisition (producers) and data processing (consumers) further complicates the optimization scope.

To handle these problems in practical IoT environments, we propose a generic system framework, named *DeepSwarm*. We define modular design for DeepSwarm and identify optimization opportunities and techniques to deploy swarm DL on resource-limited and decentralized mobile and embedded platforms. DeepSwarm pinpoints a set of *proactive* strategies inter- and intra-devices that contribute to a self-organized, self-adaptive, and self-evolving swarm DL system. Our main contributions are as follows.

- We present the concept of swarm DL, a new vision of DL system that integrates swarm intelligence into a network of DL-enbaled devices for proactive data acquisition and processing with bio-directional optimization.
- We propose DeepSwarm, a generic system framework to swarm DL. It decouples the concept of swarm DL into functional modules and highlight the challenges, opportunities, and strategies in designing each module upon prior *reactive* on-device DL techniques.
- We showcase two preliminary instances of DeepSwarm for practical IoT applications with on-device DL inference and adaptation to showcase its performance benefits in terms of accuracy and resource efficiency.

2 Scope and Framework

In this section, we introduce Swarm DL and a general framework, DeepSwarm, comprising two functional modules.

2.1 The Concept of Swarm DL

As mentioned above, *Swarm DL* extends *reactive on-device DL*, which focuses on resource-efficient DL given input data on individual IoT devices, to a distributed setting inspired by *proactive swarm intelligence*. Unlike other researches, Swarm DL has unique characteristics of self-organized, self-adaptive, and self-evolving. Specifically, Swarm DL adheres to a bottom-up organization, where each agent can decide whether to participate in cluster computing and evolve their own models based on heterogeneous data to adapt to more complex and diverse scenarios.

2.2 Swarm DL *vs.* Related Concepts

In current research, there are many related concepts that can easily cause confusion with Swarm DL, such as swarm intelligence, on device DL, distributed DL offloading, federated Learning, crowd Sensing, and so on. However, Swarm DL differs significantly from these concepts in essence. As shown in Fig. 2, we begin by comparing Swarm DL with other concepts, highlighting the unique characteristics.

- *Swarm DL vs. Swarm Intelligence*.

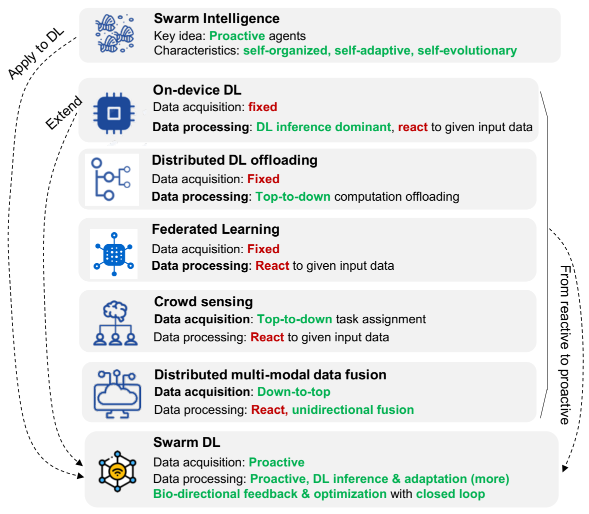

Swarm Intelligence
Key idea: Proactive agents
Characteristics: self-organized, self-adaptive, self-evolutionary
On-device DL
Data acquisition: fixed
Data processing: DL inference dominant, react to given input data
Distributed DL offloading
Data acquisition: Fixed
Data processing: Top-to-down computation offloading
Federated Learning
Data acquisition: Fixed
Data processing: React to given input data
Crowd sensing
Data acquisition: Top-to-down task assignment
Data processing: React to given input data
Distributed multi-modal data fusion
Data acquisition: Down-to-top
Data processing: React, unidirectional fusion
Swarm DL
Data acquisition: Proactive
Data processing: Proactive, DL inference & adaptation (more)
Bio-directional feedback & optimization with closed loop
Apply to DL
Extend
From reactive to proactive

Fig. 2: Comparison of swarm DL and related concepts.

- *Swarm DL vs. On-device DL*. In traditional on-device DL systems, inference constitutes the primary function, with minimal or negligible on-device adaptation, particularly when embedded devices are customized for specific inference tasks. For example, Lin [116] proposed a system, which can run DL model under a memory limit of 256kb. However, this compression and modification of the model is passively accepted by the device. In contrast, each agent in Swarm DL **exhibits greater *proactivity***, engaging in *both on-device DL inference and adaptation*, with **a higher proportion dedicated to adaptation**. Consequently, balancing resource allocation for DL inference and adaptation becomes a novel challenge in this new context.
- *Swarm DL vs. Distributed DL offloading*. Distributed DL offloading primarily involves *top-down* task partitioning and offloading to multiple devices, as highlighted in existing literature [29–31]. Swarm DL emphasizes the *bottom-up* emergence of collective behaviors among devices. In other words, in swarm DL, each agent achieves dynamic optimization of global system performance and resource efficiency by ***proactively* initiating local operations and adjustments**, rather than centralized analysis, for data acquisition and processing.
- *Swarm DL vs. Federated Learning*. Federated Learning (FL) is a kind of data processing paradigm. FL on embedded devices can boost data-driven DL training/adaptation efficiency *given input data* [32–34]. Swarm DL concentrates on the joint optimization spanning data acquisition and processing.
- *Swarm DL vs. Crowd Sensing*. Crowdsensing encompasses two categories: active and passive, depending on *human participation* [109]. Tasks can be assigned *top-to-down* based on location, tasks, or social networks to personal devices and infrastructure. It includes both real-time as well as offline data processing. In contrast, Swarm DL is a **real-time, human-independent** system composed of autonomous mobile & embedded devices (*e.g.*, smartphones and robots) that proactively acquire, process, and learn from data.
- *Swarm DL vs. Distributed multi-modal data fusion*. Distributed multi-modal data fusion focuses on integrating data from multiple sensing sources to enhance data efficiency [110], which is a *many-to-one* mode. In contrast, swarm DL *proactively* optimizes both data acquisition (supply) and processing (demand) at *each device* using ***bi-directional feedback* with *many-to-many* mode.**

In summary, Swarm DL empowers both *proactive* data acquisition for DL and data processing by DL with a closed system loop, enabling the resource-efficient DL system across a network of AIoT devices. This paradigm pushes the upper bound of the tradeoff between performance and resource efficiency beyond that of individual phases or devices set.

2.3 The DeepSwarm Framework

To realize the vision of swarm DL, we present a generic system framework, named DeepSwarm (see Fig. 3). It functions with heterogeneous AIoT hardware (*e.g.*, CPU, GPU, or MCU-equipped embedded devices), adapts to dynamic application contexts (*i.e.*, data distribution drifts and runtime resource availability), and generalizes to various AIoT applications (*e.g.*, mobile health, smart homes, autonomous vehicles, industrial automation).

DeepSwarm takes a modular design and decomposes the requirements of the swarm DL into two functional modules: *proactive swarm data acquisition for DL* and *proactive swarm data processing by DL*. These two modules work in synergy with bi-directional feedback to optimize the system performance (*e.g.*, accuracy, latency) and resource efficiency (energy efficiency, memory fragmentation). We briefly explain the scope and characteristics of the two modules below.

- *Proactive data acquisition* module leverages sensor-rich, resource-constrained AIoT devices as *agents* to form a *self-organized* swarm for efficient data acquisition, which significantly enhances perception performance and reduces redundancy among connected agents. As depicted in Fig. 3, data acquisition in DeepSwarm should autonomously select and associate agents and sensor modalities, and configure sensing parameters

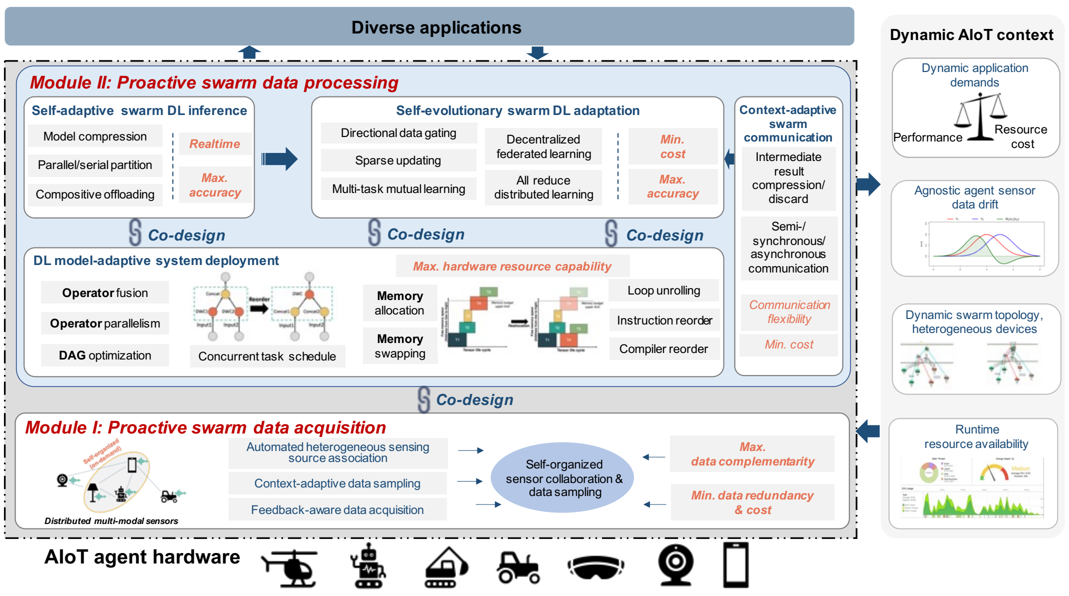

Diverse applications
Dynamic AIoT context
Module II: Proactive swarm data processing
Self-adaptive swarm DL inference
Model compression
Parallel/serial partition
Compositive offloading
Realtime
Max. accuracy
Self-evolutionary swarm DL adaptation
Directional data gating
Sparse updating
Multi-task mutual learning
Decentralized federated learning
All reduce distributed learning
Min. cost
Max. accuracy
Context-adaptive swarm communication
Intermediate result compression/ discard
Semi-/ synchronous/ asynchronous communication
Communication flexibility
Min. cost
Co-design
DL model-adaptive system deployment
Max. hardware resource capability
Operator fusion
Operator parallelism
DAG optimization
Concurrent task schedule
Memory allocation
Memory swapping
Loop unrolling
Instruction reorder
Compiler reorder
Module I: Proactive swarm data acquisition
Distributed multi-modal sensors
Automated heterogeneous sensing source association
Context-adaptive data sampling
Feedback-aware data acquisition
Self-organized sensor collaboration & data sampling
Max. data complementarity
Min. data redundancy & cost
Dynamic application demands
Performance
Resource cost
Agnostic agent sensor data drift
Dynamic swarm topology, heterogeneous devices
Runtime resource availability
AIoT agent hardware

Fig. 3: Illustration of DeepSwarm, a generic system framework to realize bio-directional optimization of Swarm DL.

such as sampling rates to maximize fused data quality with minimal overhead, such as communication and energy costs, based on data processing feedback.

- *Proactive data processing* module efficiently processes acquired data in the form of DL tasks, including inference, training, and adaptation, to enable *self-adaptive* and *self-evolving* swarm systems. Data processing in DeepSwarm is expected to scale on-demand, in real-time, with high resource efficiency.

The bi-directional optimization of two modules introduces novel co-design opportunities among research threads on swarm intelligence [43–45], on-device DL [24, 46, 47], and distributed systems [29–31], as we will elaborate in Sec. 3. We first briefly describe the functions of these modules next.

2.4 Proactive Swarm Data Acquisition Module

This module coordinates the *self-organization of distributed agents and sensors*, drawing inspiration from swarm intelligence. Each agent actively engages in data resampling, sensing parameter adjustment, and association with other agents by analyzing information extracted from cross-modal, cross-task, and cross-clock sensor data, aiming to *maximize fusion quality* and *minimize redundancy* of agent data. Additionally, it aims to achieve complementary enhancement in the early stage, addressing challenges such as modal information loss, clock asynchrony, and heterogeneous data shifts. The *early stage* refers to the initial phases of data generation, pre-processing, streaming out, and processing. This mitigates subsequent resource costs (*e.g.*, sampling, computing, transmission bandwidth) as early as possible. Specifically, we emphasize the simultaneous assessment of the *explicit* and *implicit* importance of data on the performance of subsequent data processing tasks at runtime.

- *Explicit data importance* profiling can be realized using techniques such as data-/feature-level information comparison and historical fitting.
- *Implicit data importance* profiling depends on its complementarity in the data acquisition phase across the swarm system and the characteristics of subsequent processing tasks. This includes asynchronous feature disentangling, generation, and composition, slow data repurposing, heterogeneous data distribution clustering, as well as spatio-temporal data reuse.

The explicit data importance estimation is data-driven and has abundant existing works. While implicit data complementarity profiling is system-driven and non-trivial, requiring a comprehensive consideration of dynamic system factors. We specify three main requirements essential to achieve such swarm data acquisition.

2.4.1 Automated Association of Heterogeneous, Asynchronous, and Asymmetric Sensing Sources

The distributed collaboration of data-rich sensors, such as cameras, microphones, LIDAR, and RF imagers, across diverse agents, has facilitated high accuracy in complex environments like monitoring around corners [48], through walls [49], in low light [28], and under foliage [50]. To optimize overall sensing sensitivity and coverage within resource-constrained AIoT agents, we intend to minimize data redundancy and leverage early-stage correlations among sensor modalities and agents. However, practical implementation faces several challenges:

- **Misaligned sampling rates** can occur when sensor data streams from different agents operate at different rates. For example, one agent may capture videos at 60 frames per second (fps), while another agent monitors the same environment at 30 fps, causing misalignment problems.
- **Incomplete real-time data** can be from insufficient or missing data in certain modalities, intermittent connectivity, or malfunctioning sensors. For instance, in satellite imaging, data gaps may arise due to the cyclical rotation of the satellite during specific periods.
- **Data asynchrony** arises from significant differences in data sizes among different modalities (*e.g.*, video vs. audio), leading to delays between streams due to wireless dynamics. For instance, video data often surpasses audio data in size [81]. Slow data streams may lead to *staleness* issues, causing increased latency if blocked or accuracy degradation if proceeding without waiting.

To tackle these challenges, efficient sensing source association is desired. Various explorations can enhance the explicit and implicit complementarity at the sensing source level:

- Precisely analyzing the correlations of raw data and disentangling the transferability of extracted features by computing an affinity matrix or constructing relationship graphs [82]. This innovative multimodal preprocessing can achieve data alignment of different modalities before fusion, contributing to adaptive fusion strategies that dynamically adjust integration based on real-time context and relevance.
- Transforming irrelevant data into valuable assets, such as repurposing data initially deemed irrelevant for Task A to be valuable for Task B. Effective utilization of lagging data streams, instead of directly discarding them, is crucial, with efforts focused on minimizing the impact of data staleness on accuracy.
- Achieving asynchronous complementarity involves efficiently processing slow/fast and non-synchronized data streams from multiple modalities, necessitating swift decisions and adaptations. This entails two key aspects: rapid and accurate performance prediction before execution and computation acceleration.

2.4.2 Context-adaptive Data Sampling

In a swarm of agents, adapting sampling strategies of selected sensors to the dynamics and diversity of sensing tasks is crucial for capturing necessary information without data redundancy [51]. Implementing context-aware data sampling involves acknowledging that not all classification categories necessitate high-fidelity data across every modality. Observations suggest that only a limited number of categories are sensitive to processing with partially missing data [19]. For instance, when optimizing DL model adaptation tasks, the active sample identification strategy for backward propagation should consider two criteria: *i)* identifying reliable samples for adaptation and *ii)* ensuring non-redundancy, especially across modalities and tasks. For DL inference, duty-cycle sampling for distributed swarm sensors extends battery life by predicting potential correlations after global information fusion from a long-range dependency perspective. This involves locally executed sampling, followed by sensor shutdown during idle periods or reduced sampling rates based on complementarity and environment complexity [8].

2.4.3 Feedback-aware Data Acquisition

Adapting sensing source association and sampling should autonomously respond to real-time data processing feedback, utilizing local information without relying on global data knowledge or exact processing results. These configurations must swiftly adjust to changing sensor data streams and application environments. Real-time performance estimation feedback is essential for optimizing the data acquisition pipeline. However, verifying the suitability of these strategies for subsequent processing tasks is challenging without exact execution. One approach, as seen in test-time adaptation, involves an additional "re-forward" module following forward and back-forward passes. The core challenge lies in accurately and timely predicting data acquisition performance without re-executing the entire data processing phase [83,84].

2.5 Proactive Swarm Data Processing Module

This module encompasses data processing tasks, including DL inference and adaptation. Traditional on-device DL systems primarily focus on inference, with minimal on-device adaptation, especially in customized embedded devices for specific tasks. In contrast, Swarm DL agents exhibit *higher proactivity*, engaging in both *inference* and *adaptation*, even with a greater emphasis on adaptation. Consequently, balancing resource allocation for DL inference and adaptation presents a novel challenge in this context. Specifically, the primary objective in swarm DL is to perform data processing tasks in a *self-adaptive* and *self-evolutionary* manner, akin to general swarm intelligence. Unlike approaches focused solely on optimizing DL algorithms, DeepSwarm also tackles system asynchrony in resource competition, varying resource availability among agents, managing peak memory usage, and optimizing tradeoffs between system delay and accuracy. We emphasize the following key requirements for this module.

2.5.1 Self-adaptive DL Inference

To process sensor data in real-time under low and dynamic resource budgets, we must adaptively compress and partition DL inference workloads, balancing accuracy and resource efficiency. Innovative integration of model compression, partition, and offloading techniques is needed to scale up/down DL inference workloads finely, at the operator, channel, branch, and layer level. Despite algorithm advancements in DL model compression or partition, they struggle to address the dynamic nature of real-world deployment. Recognizing and accommodating these changes are crucial due to varying performance demands influenced by time-varying resource availability, inference frequency, and resource contention from other applications. The self-adaptive DL inference process must operate retraining-free and cross-level spanning DL model, computation graph, operator, and memory allocation to match the dynamic contexts.

2.5.2 Self-evolutionary DL Adaptation

In real-world AIoT applications, *live data drifts* commonly challenge the accuracy-resource efficiency balance established by pre-trained DL models. Actively learning from new data and adapting the parameters of deployed DL models using limited local computing resources from local and edge agents is crucial. Parameter adaptation should be selectively initiated upon detecting significant shifts and must be both data- and resource-efficient to facilitate timely model adaptation to unforeseen sensor data and tasks. To maximize the proportion of high-precision DL inference time to the entire life cycle (including inference and adaptation time) for all agents, an efficient mechanism is needed to support resource-limited agents handling DL adaptation tasks. To realize this, agents must accurately and swiftly estimate DL model accuracy drops locally without ground-truth labels to control the size of retraining data and decide when to trigger model adaptation. Updating DL models to adapt to dynamic data distributions is not an easy task, but real-time updates can alleviate challenges such as limited new data, learning conflicts, and accumulation of pseudo label errors [41]. DeepSwarm also needs to balance limited computing/memory resources among agents and address the competition between asynchronous tasks initiated by different agents in a decentralized manner. Additionally, addressing operator support issues for DL training on specific embedded SoCs is crucial [42].

Note that we primarily focus on *data processing*. Accounting for properties and constraints of *data communication* among agents (*e.g.*, unreliable, or asynchronous) in DeepSwarm is also beneficial, as we will discuss in Sec. 4.

3 Challenges and Opportunities

DeepSwarm relies on highly automated and closed-loop bi-directional optimization, emphasizing interactions among the environment, heterogeneous system components, and design stacks. As shown in Fig. 4, unlike traditional research focusing on separate optimization of data acquisition and processing, the bi-directional optimization of these components enhances performance limits, balancing accuracy and resource efficiency. By jointly optimizing swarm data acquisition and processing, data can be accurately collected and processed, maximizing swarm performance while minimizing redundancy and resource costs. In IoT scenarios, each agent serves as both the *proactive data producer and consumer*, leading to intricate dynamic matching between them. New data processing tasks constantly emerge, competing for swarm agents' computing/memory resources, necessitating continuous redistribution of data and adjustments of DL model structures and resource allocation to meet diverse application requirements such as accuracy, latency, and resource efficiency. Embracing this co-design principle, we identify the following challenges and opportunities:

3.1 Self-adaptive Non-blocking DL Inference

To strike a balance between inference accuracy and efficiency, it is crucial to enhance complementarity and minimize redundancy in data acquisition and processing.

3.1.1 Identifying Non-redundant Data Correlation

To identify minimal data requirements and non-redundant data correlation, **commonly used methods** involve *statistical* analysis or *divergence measures*, such as correlation, mutual information, variance, and relative importance, to assess

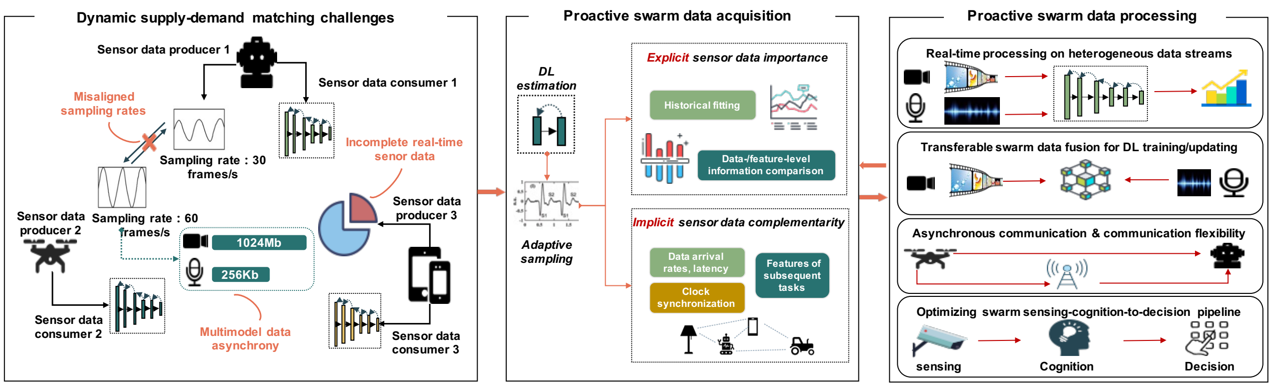

Dynamic supply-demand matching challenges
Sensor data producer 1
Sensor data consumer 1
Misaligned sampling rates
Sampling rate : 30 frames/s
Incomplete real-time senor data
Sensor data producer 2
Sampling rate : 60 frames/s
Sensor data producer 3
1024Mb
256Kb
Sensor data consumer 2
Multimodel data asynchrony
Sensor data consumer 3
Proactive swarm data acquisition
DL estimation
Adaptive sampling
Explicit sensor data importance
Historical fitting
Data-/feature-level information comparison
Implicit sensor data complementarity
Data arrival rates, latency
Features of subsequent tasks
Clock synchronization
Proactive swarm data processing
Real-time processing on heterogeneous data streams
Transferable swarm data fusion for DL training/updating
Asynchronous communication & communication flexibility
Optimizing swarm sensing-cognition-to-decision pipeline
sensing
Cognition
Decision

Fig. 4: Illustration of bio-directional optimization loop between swarm data acquisition and processing in DeepSwarm.

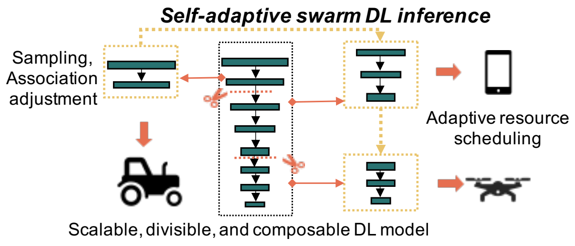

Self-adaptive swarm DL inference
Sampling, Association adjustment
Adaptive resource scheduling
Scalable, divisible, and composable DL model

Fig. 5: Illustration of self-adaptive non-blocking DL inference with heterogeneous data.

the relationships between each modality and the target variable [9]. However, existing approaches, aimed at ensuring prediction precision, are often **time-consuming** or **depend on a complete global data distribution analysis**. It is crucial to further adapt these approaches for real-time and decentralized IoT scenarios with multiple distributed agents. As shown in Fig. 5, in a distributed AIOT scenario, different devices may bring heterogeneous data for non-blocking model inference Moreover, the dynamic nature of sensing tasks in IoT scenarios adds complexity [85]. Specifically, identifying minimal data requirements needs to consider the varying relevance of different modalities based on specific tasks. *Adaptive approaches* are necessary to dynamically adjust to task dependencies.

We offer several **insights** into these challenges:

- Not every sample of sensor data contributes uniformly to the subsequent model processing accuracy. Even different classes within the same task may exhibit varying levels of impact by data. Therefore, a fine-grained task-aware estimation of data contributions is essential.
- In the DL training tasks, mishandling unreliable data samples can negatively impact accuracy. For example, samples with high entropy may introduce noisy gradients, potentially disrupting the model's stability. Even similar learning objectives may have notable parameter differences in a new epoch with fresh data due to the model's permutation invariance regarding hidden units.
- The lagging problem in data transmission in rapidly changing perception environments can also cause computational accuracy interference. Specifically, in asynchronous distributed DL learning, such increased staleness results in more significant parameter errors during DL model updates [86].
- Excluding unreliable data samples leaves potential redundancy. For instance, two similar samples with lower prediction entropy require individual gradient backpropagation, causing inefficiencies. Leveraging distinct gradients for filtering is proposed [87]. However, storing gradients for comparison becomes computationally and memory-intensive; instead, using an average value is efficient. Consequently, establishing an active sample selection criterion aimed at identifying reliable and non-redundant samples is necessary yet challenging.

3.1.2 Runtime Sampling Rate Adaptation

Adapting the sampling rate and on/off of each sensor in diverse agents based on their contribution and data processing efficiency is a non-trivial task. This challenge arises due to the dynamic nature of tasks and the varying relevance of sensors for different activities, whose difficulty is compounded by the fact that the contribution of each sensor can fluctuate based on various system factors at hand. This further makes it prohibitive to pre-determine the one-for-all sampling rate and usage, necessitating a real-time adaptive approach [10]. However, the challenge lies in the current research's inability to accurately assess the contribution of each sensor and make wise runtime adaptation decisions based on constantly changing contextual requirements, ensuring the optimal balance between data accuracy and processing efficiency in different scenarios and tasks. Factors such as sensor noise, outliers, or unexpected events can introduce complexities in algorithm design, requiring robust solutions that perform well in diverse and dynamic scenarios.

Moreover, in scenarios involving multiple agents with different sensors, the challenge lies in achieving coordination and collaboration for real-time sampling rate adaptation, despite some preliminary research efforts. For example, Wang *et al.* proposed a method [67] to optimize data sampling by balancing energy consumption, accuracy, and latency. Guan *et al.* introduced PERM [66], an optimization method to enhance the sampling efficiency of multi-path video streams. However, none of these methods have collaboratively optimized data adaptive sampling and model inference efficiency, nor have they specifically targeted optimization for real-world sensor data streams. For DL training tasks, Ren *et al.* explored the impact of sampling methods on uneven data distribution in long-tailed datasets on model training efficiency [64]. Tranheden *et al.* proposed Dacs [63], which uses cross-domain mixed sampling to perform unsupervised domain transfer. However, fundamental issues persist in these methods, such as adaptive sampling for long-tailed datasets. Current research can only mitigate bias in the model training process but cannot fully address it. This phenomenon is challenging to avoid in practical sensor data. To achieve real-time domain transfer using agnostic live sensor data, gaining deeper insights into the universality and resource efficiency of unsupervised adaptive sampling training methods is crucial. For DL evolving tasks, Brookes *et al.* proposed a robust system design method based on data-adaptive sampling [69]. Huijben *et al.* utilized Deep Probabilistic Subsampling (DPS) [71] for task-aware sampling, optimizing an optimal subset of samples with a subsequence model tailored to a specific task. However, with various tasks for swarm agents, suitable data sampling methods need to be designed across them. Current research always explores the relationship between data adaptive sampling and model adaptation from a single device, lacking a unified framework to guide it across collaborative swarm agents and diverse tasks.

3.1.3 Efficient DL Inference on Incomplete Data Stream

Real-time and precise processing of distributed multi-modal data streams is paramount for various AIoT applications. To accomplish this objective, it is crucial to investigate efficient inference frameworks capable of performing DL inference using *partial, delayed, incomplete, or missing data streams* from diverse agents. While distributed multi-modal sensing data has the potential to enhance perception accuracy and coverage through stereoscopic multi-perspective views, as observed in vehicular networks and drone clusters collaborating to sense small, occluded, or low-light targets, these challenges are common in real-world applications. This necessitates the exploration of several key technologies:

- Estimate and impute missing values and uncertainties of real-time data streams, such as employing generative models or interpolation techniques, enabling the model to make informed predictions in the absence of complete information. This enables the model to generate speculative inferences based on available data and make accurate predictions even with partial information. However, determining whether imputing missing data points can maintain the accuracy of underlying patterns, particularly without labels, presents a challenge.
- Integrate uncertainty estimation, *e.g.*, probabilistic models or Bayesian, into the DL model to quantify the uncertainty associated with predictions made on incomplete data. Adopt adaptive feature selection mechanisms to prioritize and utilize the most informative features from complete data, optimizing the fusion process.
- Develop pre-processing techniques for accurately synchronizing and aligning multi-modal data streams is crucial for accurate fusion and DL inference. However, challenges in the above techniques will arise due to *varying data arrival rates, latency, and clock synchronization* among different sensors or data sources. Moreover, sensing environments are dynamic, and the characteristics of data streams may change over time. Building adaptive models that can continuously adjust to dynamic conditions is crucial for the robustness of speculative inference in asynchronous data streams [88].
- To execute the complex computations mentioned above, especially in lightweight and accelerated implementations of generative and predictive models, is essential for real-time systems with resource-constrained agents [89].

3.1.4 Dynamically Elastic DL Model Structure

The *context-aware specification of the Swarm DL model* takes into account input sensor data, desired application performance outcomes, and the constraints of available hardware resources. It is dual-adaptive to the specific tasks and crafted to be hardware resource-friendly. Specifically, the *context-aware DL models* in inference, training, and adaptation tasks should be dynamically *scalable, divisible, and composable*.

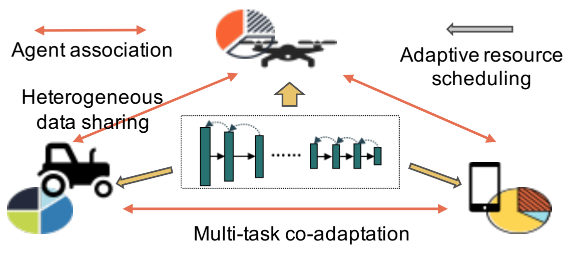


Fig. 6: Illustration of test-time swarm DL adaptation.

- The DL model, encompassing structure and parameter size, must *scale up/down* with varying compression degrees to meet dynamic resource constraints (memory, computing, and battery) and specified performance requirements (accuracy, latency, energy cost). Poorly designed DL models may result in accuracy fluctuations.
- The *divisibility* and *composability* of DL models and computation tasks are essential for distributing computation across AIoT devices. *Divisibility* depends on internal dependencies within DL layers, channels, and operators, while *composability* allows adjusting the system to offload different DL block combinations to diverse agents based on dynamic resource availability. Current algorithm-system co-design approaches, such as PCONV [100] and CoCoPIE [101], utilize fixed resource-friendly patterns to guide DL model design and operator/memory scheduling. However, they lack synergistic optimization of acquired data-aware DL model computation and system deployment based on runtime resource availability. The decentralized computation constraints, specifically the computation from partial information, further complicate optimization challenges in the co-designed swarm DL systems. Additionally, propagating runtime hardware resource availability and system execution feedback to data processing algorithm design remains a challenging aspect.

3.2 Test-time Self-evolutionary DL Adaptation

To maintain DL inference accuracy against non-stationary data drift while satisfying data acquisition and processing resource limitations, exploring collaborative learning and adaptive aggregation for DL adaptation in a swarm is necessary [11]. Specific issues include:

3.2.1 Swarm DL Adaptation with Heterogeneous Data

To achieve efficient and effective swarm DL adaptation, it is essential to consider the heterogeneity of agents in terms of data and device resources. Specifically, for each agent requiring the aggregation of multi-source data, *sensor data exhibits asynchrony* due to differences in modality, size, and transmission bandwidth, introducing asynchronous data streams. The **asynchronous nature** of distributed multi-modal data streams **poses challenges**, leading to system delays (if waiting for slow devices) or a decrease in accuracy due to insufficient or missing data modalities (if not waiting for slow data). Additionally, when each agent collaboratively processes such data, the diversity in computational resources leads to variations between fast and slow devices. The *dual system asynchrony* complicates the balance between accuracy and latency (waiting for slow devices/data introduces delays, while discarding slow devices/data may result in accuracy reduction), necessitating coordinated optimization of both data acquisition and data processing. Furthermore, the distribution heterogeneity and imbalance in sensor data distribution across different agents, including non-independently and Identically Distributed (non-IID), pose more challenges to achieving the accuracy-latency balance in the swarm DL system. To tackle these challenges, several valuable systemic considerations introduce special challenges and innovation opportunities for traditional DL inference, training, and adaptation algorithms.

Take the example of the Federated Learning (FL) scheme, where the similarity of data among different agent subsets is observed. It exhibits a clustering property in the swarm agents, it can mitigate data heterogeneity and enhance DL training accuracy by jointly training agents with similar sensor data distribution. However, clustering alone cannot resolve the issue of device asynchrony, still leading to waiting times for fast agents and consequently reducing the efficiency of swarm training. Asynchronous system mechanisms offer a solution to address device heterogeneity, but current cluster-based personalization methods are tightly linked with synchronous settings. Achieving asynchronous cluster association in FL poses two significant challenges: *i)* The clustering process for real-time agent association under asynchrony of data and devices can only gather partial information, making it challenging to obtain quick and accurate clustering results. *ii)* The staleness issue induced by data and devices asynchrony can result in the abandonment of some personalized knowledge, leading to a decrease in the accuracy of learned DL models.

Exploiting the complementary nature of different modalities in DL adaptation, particularly when faced with asynchronous real-time sensor data is a promising direction. Specifically, the scarcity of high-quality sensor data introduces a bottleneck in the mutual information between single-

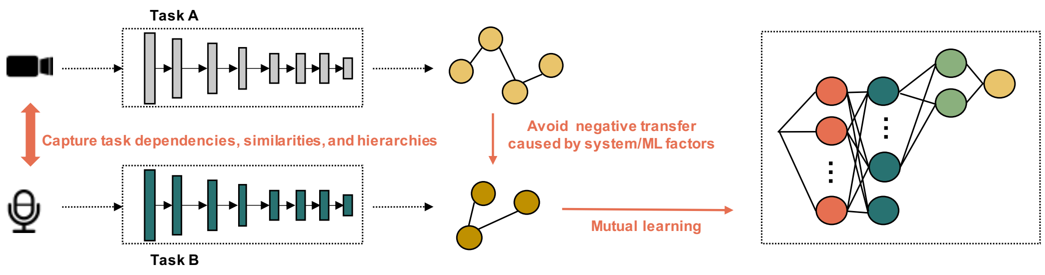

Task A
Capture task dependencies, similarities, and hierarchies
Avoid negative transfer caused by system/ML factors
Mutual learning
Task B

Fig. 7: Illustration of knowledge-enhanced multi-task co-adaptation.

modal data and downstream tasks of models [80]. Multi-modal data offers a more comprehensive representation of objects, improving the model's accuracy and robustness. Therefore, it is critical to mine the correlation between data features and target tasks, sharing the data most relevant to models' downstream tasks, to achieve complementation and enhancement of data. Also, it requires investigating fine-grained fusion strategies that effectively leverage the *unique information* present in each modality and beyond simple concatenation or early/late fusion.

3.2.2 Multi-task Co-adaptation

We explore collaborative swarm DL adaptation that considers the characteristics of diverse tasks and data. Integrating multi-task DL learning frameworks to enhance adaptation performance with limited local data and resource is an exciting direction. In different tasks, DL models generate a variety of transferable knowledge during training, part of which can augment other models' comprehension of their target tasks. Such transferable knowledge learning enhances models' feature representation and semantic understanding, enabling high-accuracy DL model training. However, this benefit is confined to task-related scenarios. In cases where task differences are obvious, multi-task learning will suffer from negative transfer [75] and seesaw phenomena [76], making accuracy gain lower than expected. Therefore, it is desired to capture task dependencies, similarities, and hierarchies for effective *knowledge transfer and representation sharing* across tasks in an asynchronous system (see Fig. 7), rather than at a fixed scheme. Additionally, within a swarm system, there might be local correlations among multiple tasks for several agents, and harnessing this local interdependence to mutual DL adaptation can be also valuable.

3.3 Adaptive System Resource Scheduling

In model-adaptive system deployment, maximizing runtime hardware capability goes beyond algorithms. Practices like intermediate result reuse and eliminating storage fragmentation are crucial for enhancing performance limits. Even with widely used DL model configurations, mapping model layers/operators onto different underlying resources in varying sequences results in diverse latency and resource overhead. For instance, addressing memory fragmentation in SqueezeNet's tensor layout can reduce wasted memory significantly. Thus, strategic optimization of the computation graph, operator parallelism/fusion, and memory allocation across processors can enhance runtime resource availability for DL execution. Moreover, algorithm-system co-design in swarm systems should facilitate iterative exploration of a more flexible DL model design space at the algorithm level, pushing the boundaries on accuracy-resource trade-offs, as shown in Fig. 8.

However, despite previous efforts in algorithm-system co-design for individual embedded devices [14], the optimization scope for the networked swarm DL system requires redefinition. Possible areas for further research include:

- Tailoring the system scheduling to the characteristics of data processing tasks, such as tensor/operator life cycles and dependencies, can enhance computation parallelism, increase data reuse, and reduce memory fragmentation during swarm data processing. We note that, unlike traditional operating systems that allocate resources for unpredictable tasks, DL models exhibit relatively fixed computation patterns. This stronger predictability in terms of computation dependencies and life cycles makes fine-grained optimizations more feasible. For example, Geoffrey *et al.* [72] leverage the repeated computation characteristics of DL models to characterize resources required for the entire training by predicting the time of a single iteration.
- While individual agents may make separate resource

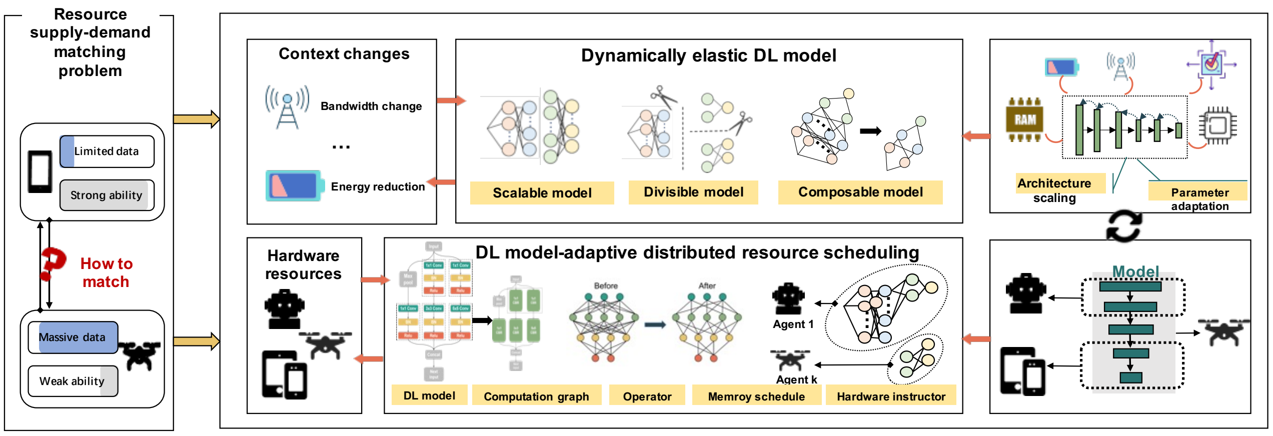


Fig. 8: Problem illustration of adaptive system resource scheduling.

pool scheduling decisions, a global performance evaluation is crucial.

Furthermore, to address the *heterogeneity* among agents, an *automated framework* is essential for a self-adaptive and self-adaptationary swarm DL system. This framework involves the fine-grained search space, pre-execution performance predictors, and schedulers. For instance, Li *et al.* optimize DL inference by searching from heterogeneous computing hardware pools with minimal online exploration costs [74]. Astra [72] significantly reduces the state space that needs to be searched in the prediction model through lightweight analysis and summary data indexing. However, these studies primarily focus on computation task offloading by predicting the deployment cost of DL models without considering how to adapt to dynamic context changes in a swarm system. *iv)* Current DL frameworks such as TensorFlow Lite [102], MNN [103], and NCNN [104] are sub-optimal for heterogeneous swarm systems. Because these frameworks, initially designed for optimizing a single DL model, may not fully leverage the capabilities of various processors in modern embedded agents. Additionally, the absence of a unified framework for deploying and evaluating DL models' deployment costs across various agents, heterogeneous CPU/GPU/NPU/DSP processors, and DL frameworks poses challenges in comparing and measuring selectable technique performance. More insights and efforts on the cross-agent framework are required.

Scheduling suitable agents within a closely connected and resource-constrained swarm is necessary yet challenging [15]. We have observed that a data processing task can only be executed if it obtains enough memory resources, and its execution time decreases as computing resources increase. Specifically, several aspects could be researched:

- It is intractable to monitor the *dynamic resource availability* with heterogeneous backends and predict the *resource requirements* of data acquisition and DL tasks. Especially, it is challenging to provide an accurate and timely estimate of DL inference performance.
- When multiple tasks run concurrently, there will be competition for shared resources among agents. Each agent can control the scheduling time for different tasks by determining when to allocate sufficient memory resources for them. It requires reasonable planning based on task urgency, computational cost, and training time. To maximize resource utilization, it is significant to match allocated resources with task requirements, which requires a full analysis of the relationship between allocated resources and latency for various tasks.
- Runtime optimizer, *e.g.*, dynamic programming, reinforcement learning, or graph search, for multi-objective optimization, is a fundamental NP-hard problem to adjust all system layers (*e.g.*, operator order, memory allocation) to achieve optimal resource supply-demand matching.
- A well-designed search space is crucial to address the attribute differences among multiple data processing demands and enhance the search speed of scheduling. It is also NP-hard to adaptively schedule all asynchronous tasks with different resource demands to fit them into the server with dynamic resource availability and maximize overall performance. Selecting task combinations can effectively reduce the number of searches, thereby improving speed. For example, we explored optimizing the search space through a task grouping mechanism that takes into account the grouping probability in the dynamic system context [105].

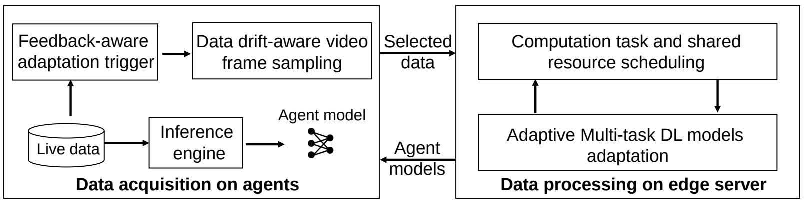

Feedback-aware adaptation trigger
Data drift-aware video frame sampling
Live data
Inference engine
Agent model
Data acquisition on agents
Selected data
Agent models
Computation task and shared resource scheduling
Adaptive Multi-task DL models adaptation
Data processing on edge server

Fig. 9: Illustration of proactive data acquisition and processing for DL adaptation tasks.

3.4 Asynchronous and Flexible Communication

In a decentralized swarm, effective communication among agents plays a crucial role in information sharing [90], cooperative sensing [91], exchanging model updates [92], and collaborating on decision-making [12]. However, the distributed data acquisition and decentralized data processing context pose challenges in terms of scalability and efficiency. For example, naively transmitting raw sensor data consumes excessive bandwidth, *e.g.*, full frames of high-fidelity data could exceed 300 Mbps, rendering it unsustainable even with advanced wireless communication technologies [93–95]. Thus, data pre-processing for selective transmission is essential but challenging. Particularly, the inter-agent *time gaps* from different agents, in data fusion, will result in inaccurate merging and compromise the performance of subsequent sensing tasks. We define the data arrival time gaps as the time between data being generated by the producer and reaching the consumer. This period involves sensor data *pre-processing, transmission, and post-processing*, delaying subsequent tasks. Therefore, the key opportunity and challenge is tuning these three system components to balance computation cost and intermediate data size under *dynamic data complexity, local resource availability, and communication bandwidth* to minimize overall time gaps. Also, it becomes necessary to consider the flexibility of asynchronous and synchronous communication for balancing immediate coordination and accuracy [96].

4 Case Study

We showcase the advantages of swarm DL with two preliminary studies, *i.e.*, swarm DL adaptation and swarm DL federated learning (FL).

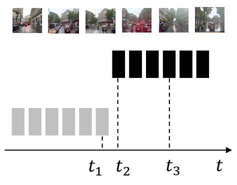
4.1 DeepSwarm for Swarm DL Adaptation

Mobile video applications today have attracted significant attention. The compressed DL model is widely used to enable on-device video inference and analysis.however, the accuracy of inference is vulnerable to the *non-stationary data drift* of the live video captured from dynamically changing mobile environments. To combat data drifts, proactive and continuous adaptation of DL models at each swarm agent with freshly collected data is desired.

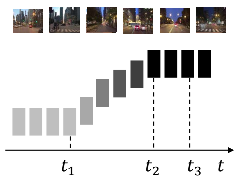
We present a swarm DL adaptation system based on the DeepSwarm framework, which enables each agent to continuously update using newly collected sensor data from local and other agents, as shown in Figure 9. Multiple agents can independently initiate this DL adaptation task. The primary objective is to maximize the average accuracy, representing the ratio of high-accuracy inference time to the total life cycle (including inference, waiting, and retraining time) for all agents. It consists of the following three modules.

4.1.1 Data Drift-aware Video Frame Sampling

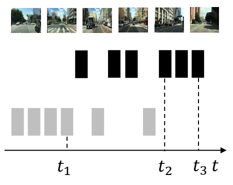
Acquiring appropriate data from swarm agents is crucial for the adaptation of DL models. Previous work [11, 77, 78] primarily retrains on-device DL models using data from a single agent, which results in low model generalization and diminished accuracy over time. Real-time data from a single device is limited, while data sharing across devices can supplement each other, enrich the DL model's representation, and improve accuracy and generalization in downstream tasks. To achieve rapid DL adaptation with minimal video frames and maximized accuracy gain at each agent, we adopt a dedicated frame sampling strategy in a data drift-aware manner for various data drifts, as illustrated in Fig. 10. In IoT environments, we categorize live sensor data drift into three different types, *i.e.*, *sudden*, *incremental*, and *gradual* drifts. For sudden or incremental drifts, we use a linear sampling rate to efficiently select video frames, ensuring significant data for adapting the DL model. For gradual drifts specifically, we start with the frame difference method to eliminate redundant frames. Subsequently, we assess the distance between the non-redundant frame and the global view of the data distribution to predict the contribution of every frame and select the most representative frames for subsequent DL adaptation tasks.



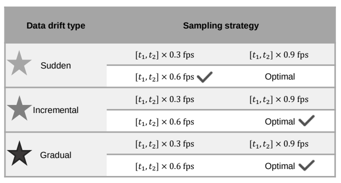
(a) Sudden drift



(b) Incremental drift



(c) Gradual drift



Data drift type	Sampling strategy	
Sudden	$[t_1, t_2] \times 0.3$ fps	$[t_1, t_2] \times 0.9$ fps
	$[t_1, t_2] \times 0.6$ fps ✓	Optimal
Incremental	$[t_1, t_2] \times 0.3$ fps	$[t_1, t_2] \times 0.9$ fps
	$[t_1, t_2] \times 0.6$ fps	Optimal ✓
Gradual	$[t_1, t_2] \times 0.3$ fps	$[t_1, t_2] \times 0.9$ fps
	$[t_1, t_2] \times 0.6$ fps	Optimal ✓

(d) Sampling strategies

Fig. 10: Different types of sensor data drift and the corresponding adaptive sampling rate on these drift types.

Table 1: Performance comparison of different DL model adaptation methods

Method	**Accuracy gain of global model**	**Accuracy of mobile model after adaptation**				**Accuracy gain of mobile model**			
	IoU = 0.5	IoU = 0.50				IoU = 0.50			
		Mobile model A	Mobile model B	Mobile model C	Average	Mobile model A	Mobile model B	Mobile model C	Average
Domain adaptation	None	0.504	0.469	0.497	0.49	14.3%	48.9%	13.7%	23.4%
NestEvo without data generation	1.3%	0.504	0.475	0.501	0.505	14.3%	50.8%	15.5%	27.2%
Original mobile model	None	0.441	0.315	0.437	0.397	None			
Only mobile model adaptation	0	0.501	0.478	0.493	0.491	13.6%	51.7%	12.8%	23.7%
NestEvo	9.13%	0.571	0.543	0.584	0.566	29.5%	72.4%	33.6%	42.6%

4.1.2 Feedback-aware DL Adaptation Trigger

Accurately triggering the DL model adaptation at each agent can maintain model accuracy over their life cycle while minimizing system overhead. Insufficient adaptation frequency leads to prolonged periods of low-accuracy inference, while excessive adaptation frequency increases overhead at the agent. Detecting data drift time points, where DL model accuracy significantly drops, is essential for achieving accuracy and adaptive adaptation trigger. However, measuring accuracy drop feedback in dynamic mobile environments with resource limitations based on unlabeled sensor data is challenging. We employ confidence scores, encompassing classification and localization confidence, to enhance the agent's processing accuracy. Moreover, we observe that the onset of the accuracy drop is not always the optimal trigger timepoint for DL adaptation. To identify the trigger timepoint, we develop a sliding window to measure accuracy drop rate and monitor data drift. Specifically, adaptation is initiated when the drop surpasses a pre-set threshold, transitioning from the old to new data distribution. Then, agents assess confidence variance to determine if the new distribution stabilizes, enabling more tailored and effective adaptation with real-time sensor data.

4.1.3 Adaptive DL Adaptation and Resource Scheduling

Enhancing average accuracy throughout the lifecycle entails prolonging high-accuracy inference periods and minimizing retraining time. We use predicted performance metrics for each task, such as accuracy gain, training time, and resource demand, to develop optimal task scheduling strategies. We estimate memory requirements for each task by analyzing layer parameters and features. To profile training time, we build a lightweight prediction network. We also develop a non-negative least squares solver to model the relationship between accuracy and training time for predicting accuracy gains of an adaptation task. Using these metrics, we employ a dynamic programming algorithm to select asynchronous adaptation tasks and utilize in-memory computing to accelerate adaptation, reducing overall retraining latency. This scheme efficiently handles matrix multiplication for generating pseudo-labels. Concurrently, we optimize the retraining process to mitigate the impact of resistor noise on accuracy.

4.1.4 Experimental Results

We compare our system with four baselines, *i.e.*, domain adaptation [78], DeepSwarm without data fusion from other agents, original agent DL model, and single-agent adaptation [79]. Specifically, domain adaptation is an effective method to deal with data drift using adversarial training. DeepSwarm without data fusion from other agents and single-agent adaptation baselines both employ knowledge distillation with single-source data for adaptations, but the former includes the global model updating. In this experiment, we employ three types of agents with lightweight DL models, *i.e.*, pruned Faster-RCNNs with accuracies of 0.441,

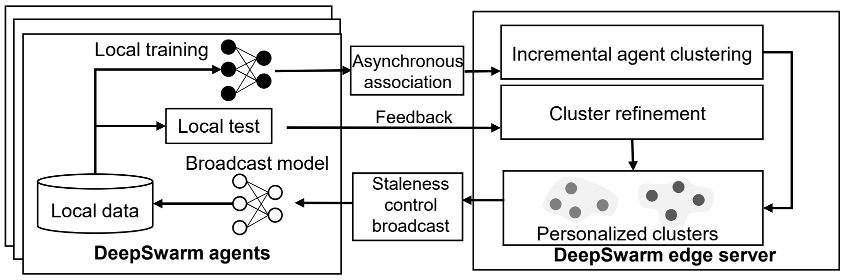


Fig. 11: Illustration of proactive data acquisition and processing for asynchronous personalized FL.

0.315, and 0.437. Using videos collected by these agents, we compare the accuracy gain of the retraind DL models. Table 1 demonstrates that DeepSwarm achieves the best overall performance in terms of accuracy, enhancing average accuracy by over 40% compared to the original models for the mobile model and by 9% for the global model.

4.2 DeepSwarm for Asynchronous Personalized FL

We further integrate DeepSwarm with the federated learning framework in IoT systems. Federated learning (FL) allows agents upload DL models instead of transmitting data for joint DL training. However, real-world mobile systems with asynchronous sensor data and heterogeneous agent resources pose challenges to FL. Traditional synchronous FL methods often result in significant delays and inefficiencies due to varying upload times for models. To address this, we transition to asynchronous FL. Another challenge is personalization, as mobile agents exhibit significant variations in bias toward sensing data and DL inference results.

Utilizing *self-organized agent associations* to train multiple personalized DL models can enhance inference accuracy. However, existing personalized FL methods assume synchronous, which is impractical. Achieving real-time self-organized agent association and asynchronous personalized FL with high accuracy and efficiency poses significant challenges for the swarm DL system. In addition, highly asynchronous sensor data and heterogeneous devices can cause delays in DL model uploading from agents, known as the *staleness problem*. This leads to errors in the personalized DL model aggregation process in FL, reducing accuracy. Therefore, we employ DeepSwarm for staleness control in asynchronous personalized federated learning of swarm DL models. DeepSwarm comprises two main technical blocks.

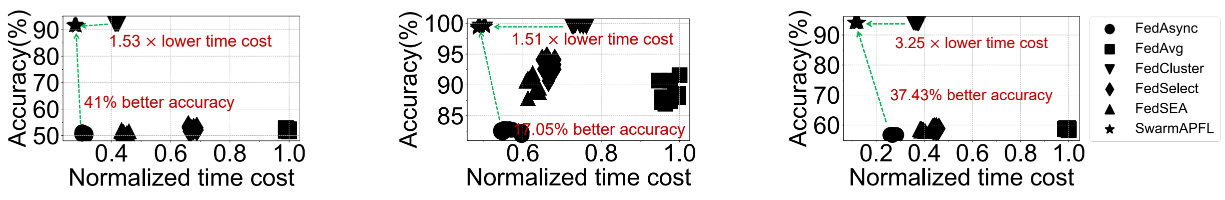
4.2.1 Data-aware Asynchronous Agent Association for Staleness Control

To enable efficient asynchronous personalized FL among swarm agents with diverse data distribution and hardware resources, DeepSwarm facilitates rapid agent association and mitigates staleness issues stemming from asynchronous sensor data and agents. For real-time agent clustering, DeepSwarm employs an on-arrival initial cluster association mechanism. As updates to the DL model arrive asynchronously, clusters are initialized gradually. Initially, cluster centers are set using the first few arriving model parameters at the server, continuing until the predefined cluster number threshold is reached. Then, the server calculates the L1 distance that is simple to quantify the difference between newly arrived DL model parameters and existing cluster centers' model parameters. Agents are then assigned to the closest cluster based on the minimum L1 distance.

To control model staleness resulting from asynchronous agents in the swarm, we present to promptly distribute DL models newly aggregated by the server to agents within the cluster. DeepSwarm defines the personalized cluster as the broadcast range. While data heterogeneity can pose challenges in traditional FL, it's beneficial in personalized FL. Heterogeneous data aids models in adapting to specific scenarios or tasks, enhancing model personalization. By restricting broadcasts to in-cluster clients with similar data distributions, we narrow the broadcast scope. In-cluster broadcasting occurs on relatively homogeneous data, enhancing tolerance to staleness and potentially reducing broadcast frequency. Broadcast frequency is dynamically adjusted based on historical model changes since the last broadcast and predicted changes before the next aggregation. Broadcast is triggered when predicted changes exceed historical changes.

4.2.2 Feedback-aware Dynamic Cluster Refinement

Although initial clusters capture agent associations, errors may occur. Agents simultaneously evaluate the effectiveness



(a) CIFAR-10 (b) HAR (c) Ubisound

Fig. 12: Comparison of accuracy *vs.* training time between DeepSwarm and other baselines on diverse tasks.

of the assigned cluster center model on their local sensor data and provide this *feedback* to the server. Throughout the FL process, the server dynamically refines clusters based on uploaded feedback to fine-tune clustering. Cluster refinement involves two iterative steps: expansion and merging.

- Low feedback from an agent indicates inaccurate agent collaboration. In response, the server initiates an clsuter *expansion* process, assigning the agent to a new cluster. To prevent overfitting in newly expanded clusters with few agents, most parameters in the center model are frozen, and only the output layer is trained exclusively. The freezing constraint is gradually lifted until the server triggers cluster merging.
- Cluster *merging* occurs when the number of clusters exceeds a threshold. Clusters with similar data distribution are iteratively combined until the number of clusters drops below the merging threshold. However, as model parameters of different cluster centers are trained separately, they may fit diverse and potentially conflicting data distributions. Simple aggregation of their parameters as the center of the merged cluster would be suboptimal. Thus, we adopt a weight aggregation scheme based on fine-grained momentum.

4.2.3 Experimental results

We compared DeepSwarm with five SOTA baselines, including the standard FL method FedAvg, the asynchronous FL method FedAsync, the semi-asynchronous FL method FedSelect, and FedSEA, as well as the personalized FL method ClusterFL. We conducted experiments to validate the effectiveness of DeepSwarm across three typical tasks, including image classification (IC), human activity recognition (HAR), and sound detection (SD). In this experiment, we utilize a simulation environment where our simulated cluster comprises 20% Jetson Nano, 20% Jetson NX Xavier, 20% Jetson Orin, and 40% Raspberry Pi 4. For IC, we divide CIFAR-10 [114] into 120 clients. For HAR, we split UCI-HAR [115] into 30 clients. For SD, we split Ubisound [8] into 10 clients. The results showed that DeepSwarm achieves a reduction of up to 88.2% in convergence time, and an improvement of up to 46% in accuracy.

5 Conclusion

In this paper, we introduced the concept of Swarm DL, extending existing on-device DL paradigms that predominantly rely on *predefined* data processing patterns to *react* given data. This reliance often leads to accuracy and resource efficiency bottlenecks. Inspired by the collective intelligence observed in natural swarms, where individual *proactive* actions contribute to superior global performance, we advocate for a shift towards *Swarm DL*. By harnessing the potential of physically adjacent mobile and embedded devices in IoT scenarios, we present DeepSwarm, a closed-loop system framework architecture. DeepSwarm facilitates bidirectional optimization between data acquisition and processing, aiming to push the performance boundaries of on-device DL Specifically, DeepSwarm addresses the requirements of proactive swarm DL by decomposing them into layers: self-organized swarm data acquisition and self-adaptive, self-evolutionary swarm data processing. We discuss the challenges of decoupled functional modules and present two preliminary case studies of DeepSwarm to demonstrate its advantages. In the future, DeepSwarm still encounter challenges such as dynamics and diversity in data, resources, and performance demands, runtime and automatic bio-optimization will be crucial for swarm DL's future development.

Acknowledgements This work was partially supported by the National Science Fund for Distinguished Young Scholars (62025205), the National Natural Science Foundation of China (62032020, 62102317), and CityU APRC Grant (9610633).

References

1. Liu S, Guo B, Fang C, et al. Enabling Resource-Efficient AIoT System With Cross-Level Optimization: A Survey. IEEE Communications Surveys & Tutorials, 2023.
2. Kruzan K P, Ng A, Stiles-Shields C, et al. The perceived utility of smartphone and wearable sensor data in digital self-tracking technologies for mental health. In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems. 2023: 1-16.
3. Ma S, Pei J, Zhang W, et al. Neuromorphic computing chip with spatiotemporal elasticity for multi-intelligent-tasking robots. Science Robotics, 2022, 7(67): eabk2948.
4. Hu Y, Fang S, Lei Z, et al. Where2comm: Communication-efficient collaborative perception via spatial confidence maps. Advances in neural information processing systems, 2022, 35: 4874-4886.
5. Bonabeau E, Dorigo M and Theraulaz G. Swarm intelligence: from natural to artificial systems. Number 1. Oxford university press, 1999.
6. Jia F C, Zhang D Y, Cao T, et al. CoDL: efficient CPU-GPU coexecution for deep learning inference on mobile devices. In: Proceedings of MobiSys, Oregon, 2022. 209–221
7. Qu Z, Zhou Z, Tong Y, et al. p-meta: Towards on-device deep model adaptation. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2022: 1441-1451.
8. Liu S C, Zhou Z M, Du J Z, et al. UbiEar: Bringing location-independent sound awareness to the hard-of-hearing people with smartphones. Proceedings of IMWUT, 2017, Vol: 1–21
9. Liu C H, Zhang L, Liu Z Q, et al. Lasagna: Towards deep hierarchical understanding and searching over mobile sensing data. In: Proceedings of MobiCom, New York, 2016. 334–347
10. Khani M, Hamadanian P, Nasr-Esfahany A, et al. Real-time video inference on edge devices via adaptive model streaming. In: Proceedings of CVPR, Virtual, 2021. 4572–4582
11. Khani M, Ananthanarayanan G, Hsieh K, et al. RECL: Responsive Resource-Efficient Continuous Learning for Video Analytics. In: Proceedings of NSDI, Boston, 2023. 917–932
12. McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of Artificial intelligence and statistics, Ft. Lauderdale, 2017. 1273–1282
13. Ding Y Y, Zhu L G, Jia Z H, et al. IoS: Inter-operator scheduler for cnn acceleration. In: Proceedings of MLSys, Virtual, 2021. 167–180
14. Verbraeken J, Wolting M, Katzy J, et al. A survey on distributed machine learning, 2020, 2: 1–33
15. Mayer R, Jacobsen H. Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools. ACM Computing Surveys (CSUR), 2020, 1: 1–37
16. Bhardwaj R, Xia Z X, Ananthanarayanan G, et al. Ekya: Continuous learning of video analytics models on edge compute servers. In: Proceedings of NSDI, RENTON, 2022. 119–135
17. Zeng X, Fang B Y, Shen H C, et al. Distream: scaling live video analytics with workload-adaptive distributed edge intelligence. In: Proceedings of SenSys, Yokohama, 2020. 409–421
18. Xu J G, Cao H, Yang Z, et al. SwarmMap: Scaling up real-time collaborative visual SLAM at the edge. In: Proceedings of NSDI, RENTON, 2022. 977–993
19. Li T X, Huang J, Risinger E, et al. Low-latency speculative inference on distributed multi-modal data streams. In: Proceedings of MobiSys, Virtual, 2021. 67–80
20. Wang H, Guo B, Liu J, et al. Context-aware adaptive surgery: A fast and effective framework for adaptative model partition. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2021, Vol: 5(3): 1-22.
21. Samal K, Kumawat H, Saha P, et al. Task-driven rgb-lidar fusion for object tracking in resource-efficient autonomous system. IEEE Transactions on Intelligent Vehicles, 2021, Vol: 7(1): 102-112.
22. Yang H, Alphones A, Zhong W D, et al. Learning-based energy-efficient resource management by heterogeneous RF/VLC for ultra-reliable low-latency industrial IoT networks. IEEE Transactions on Industrial Informatics, 2019, Vol: 16(8): 5565-5576.
23. Guo B, Liu Y, Liu S, et al. Crowdhmt: crowd intelligence with the deep fusion of human, machine, and IoT. IEEE Internet of Things Journal, 2022, Vol: 9(24): 24822-24842.
24. Pang B W, Liu S, Wang H, et al. AdaMEC: Towards a Context-Adaptive and Dynamically-Combinable DNN Deployment Framework for Mobile Edge Computing. ACM Transactions on Sensor Networks, 2023.
25. Huang Y, Cheng Y, Bapna A, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. Advances in neural information processing systems, 2019, 32.
26. Rostami M, Berahmand K, Nasiri E, et al. Review of swarm intelligence-based feature selection methods. Engineering Applications of Artificial Intelligence, 2021, Vol: 100: 104210.
27. Attiya I, Abd Elaziz M, Abualigah L, et al. An improved hybrid swarm intelligence for scheduling iot application tasks in the cloud. IEEE Transactions on Industrial Informatics, 2022, Vol: 18(9): 6264-6272.
28. Liu S, Li X, Zhou Z, et al. AdaEnlight: Energy-aware Low-light Video Stream Enhancement on Mobile Devices. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2023, Vol: 6(4): 1-26.
29. Liu S, Guo B, Ma K, et al. AdaSpring: Context-adaptive and runtime-evolutionary deep model compression for mobile applications. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2021, Vol: 5(1): 1-22.
30. Zeng X, Yan M, Zhang M. Mercury: Efficient on-device distributed dnn training via stochastic importance sampling. In: Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems. 2021: 29-41.
31. Hao P, Zhang Y. Eddl: A distributed deep learning system for resource-limited edge computing environment. In: Proceedings of 2021 IEEE/ACM Symposium on Edge Computing (SEC). IEEE, 2021: 1-13.
32. Rapp M, Khalili R, Pfeiffer K, et al. Distreal: Distributed resource-aware learning in heterogeneous systems. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2022, 36(7): 8062-8071.
33. Horvath S, Laskaridis S, Almeida M, et al. Fjord: Fair and accurate

federated learning under heterogeneous targets with ordered dropout. Advances in Neural Information Processing Systems, 2021, Vol: 34: 12876-12889.

34. Wu S, Li T, Charles Z, et al. Motley: Benchmarking heterogeneity and personalization in federated learning. arXiv preprint arXiv:2206.09262, 2022.
35. Jalil B, Leone G R, Martinelli M, et al. Fault detection in power equipment via an unmanned aerial system using multi modal data. Sensors, 2019, Vol: 19(13): 3014.
36. Ahmad F, Shin C S, Ghosh R, et al. AeroTraj: Trajectory Planning for Fast, and Accurate 3D Reconstruction Using a Drone-based LiDAR. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2023, Vol: 7(3): 1-28.
37. Guerlin L, Pannetier B, Rombaut M, et al. Study on group target tracking to counter swarms of drones. In: Proceedings of Signal processing, sensor/information fusion, and target recognition XXIX. SPIE, 2020, 11423: 8-27.
38. Rashidi S, Sridharan S, Srinivasan S, et al. Astra-sim: Enabling sw/hw co-design exploration for distributed dl training platforms. In: Proceedings of 2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 2020: 81-92.
39. Chiu T C, Shih Y Y, Pang A C, et al. Semisupervised distributed learning with non-IID data for AIoT service platform. IEEE Internet of Things Journal, 2020, Vol: 7(10): 9266-9277.
40. Al-Habob A A, Dobre O A, Armada A G, et al. Task scheduling for mobile edge computing using genetic algorithm and conflict graphs. IEEE Transactions on Vehicular Technology, 2020, Vol: 69(8): 8805-8819.
41. Liu Y, Zhou W, Xi M, et al. Multi-modal context propagation for person re-identification with wireless positioning. IEEE Transactions on Multimedia, 2021, Vol: 24: 3060-3073.
42. Springer T, Eiroa-Lledo E, Stevens E, et al. On-device deep learning inference for system-on-chip (SoC) architectures. Electronics, 2021, Vol: 10(6): 689.
43. Tang J, Liu G, Pan Q. A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. IEEE/CAA Journal of Automatica Sinica, 2021, Vol: 8(10): 1627-1643.
44. Rostami M, Berahmand K, Nasiri E, et al. Review of swarm intelligence-based feature selection methods. Engineering Applications of Artificial Intelligence, 2021, Vol: 100: 104210.
45. Beni G. Swarm intelligence. Complex Social and Behavioral Systems: Game Theory and Agent-Based Models, 2020: 791-818.
46. Chen Y, Zheng B, Zhang Z, et al. Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions. ACM Computing Surveys (CSUR), 2020, Vol: 53(4): 1-37.
47. Murshed M G S, Murphy C, Hou D, et al. Machine learning at the network edge: A survey. ACM Computing Surveys (CSUR), 2021, Vol: 54(8): 1-37.
48. Zhang D, Wang J, Jang J, et al. On the feasibility of wi-fi based material sensing. In: Proceedings of The 25th Annual International Conference on Mobile Computing and Networking. 2019: 1-16.
49. Zeng Y, Wu D, Xiong J, et al. FarSense: Pushing the range limit of WiFi-based respiration sensing with CSI ratio of two antennas. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2019, Vol:3(3): 1-26.
50. Zhong Y, Bi T, Wang J, et al. A climate adaptation device-free sensing approach for target recognition in foliage environments. IEEE Transactions on Geoscience and Remote Sensing, 2022, Vol: 60: 1-15.
51. Sun T, Lu C, Ling H. Local Context-Aware Active Domain Adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 18634-18643.
52. Liu S, Du J, Nan K, et al. AdaDeep: A usage-driven, automated deep model compression framework for enabling ubiquitous intelligent mobiles. IEEE Transactions on Mobile Computing, 2020, Vol: 20(12): 3282-3297.
53. Weyns D, Gheibi O, Quin F, et al. Deep learning for effective and efficient reduction of large adaptation spaces in self-adaptive systems. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2022, Vol: 17(1-2): 1-42.
54. Chen Y, Ning Y, Slawski M, et al. Asynchronous online federated learning for edge devices with non-iid data. In: Proceedings of 2020 IEEE International Conference on Big Data (Big Data). IEEE, 2020: 15-24.
55. Tun Y K, Park Y M, Tran N H, et al. Energy-efficient resource management in UAV-assisted mobile edge computing. IEEE Communications Letters, 2020, Vol: 25(1): 249-253.
56. Tan A Z, Yu H, Cui L, et al. Towards personalized federated learning. IEEE Transactions on Neural Networks and Learning Systems, 2022.
57. Ma X, Zhang J, Guo S, et al. Layer-wised model aggregation for personalized federated learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 10092-10101.
58. You C, Feng D, Guo K, et al. Semi-synchronous personalized federated learning over mobile edge networks. IEEE Transactions on Wireless Communications, 2022.
59. Zhang W, Chen X, He K, et al. Semi-asynchronous personalized federated learning for short-term photovoltaic power forecasting. Digital Communications and Networks, 2023, Vol: 9(5): 1221-1229.
60. Baccarelli E, Scarpiniti M, Momenzadeh A, et al. AFAFed—Asynchronous Fair Adaptive Federated learning for IoT stream applications. Computer Communications, 2022, Vol: 195: 376-402.
61. Miao X, Nie X, Zhang H, et al. Hetu: A highly efficient automatic parallel distributed deep learning system. 2023.
62. Romero J, Yin J, Laanait N, et al. Accelerating collective communication in data parallel training across deep learning frameworks. In: Proceedings of 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). 2022: 1027-1040.
63. Tranheden W, Olsson V, Pinto J, et al. Dacs: Domain adaptation via cross-domain mixed sampling. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2021: 1379-1389.
64. Ren J, Yu C, Ma X, et al. Balanced meta-softmax for long-tailed visual recognition. Advances in neural information processing systems, 2020, Vol: 33: 4175-4186.
65. Perslev M, Jensen M, Darkner S, et al. U-time: A fully convolutional network for time series segmentation applied to sleep staging. Advances in Neural Information Processing Systems, 2019, Vol: 32.

66. Guan Y, Zhang Y, Wang B, et al. PERM: Neural adaptive video streaming with multi-path transmission. In: Proceedings of IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE, 2020: 1103-1112.
67. Wang C, Zhang S, Chen Y, et al. Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics. In: Proceedings of IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE, 2020: 257-266.
68. Kim D, Tsai Y H, Zhuang B, et al. Learning cross-modal contrastive features for video domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 13618-13627.
69. Brookes D, Park H, Listgarten J. Conditioning by adaptive sampling for robust design. In: Proceedings of International conference on machine learning. PMLR, 2019: 773-782.
70. Zhai Y, Lu S, Ye Q, et al. Ad-cluster: Augmented discriminative clustering for domain adaptive person re-identification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 9021-9030.
71. Huijben I, Veeling B S, van Sloun R J G. Deep probabilistic subsampling for task-adaptive compressed sensing. In: Proceedings of 8th International Conference on Learning Representations, ICLR 2020. 2020.
72. Geoffrey X Y, Gao Y, Golikov P, et al. Habitat: A Runtime-Based computational performance predictor for deep neural network training. In: Proceedings of 2021 USENIX Annual Technical Conference (USENIX ATC 21). 2021: 503-521.
73. Sivathanu M, Chugh T, Singapuram S S, et al. Astra: Exploiting predictability to optimize deep learning. In: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. 2019: 909-923.
74. Li B, Samsi S, Gadepally V, et al. Kairos: Building cost-efficient machine learning inference systems with heterogeneous cloud resources. In: Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing. 2023: 3-16.
75. Standley T, Zamir A, Chen D, et al. Which tasks should be learned together in multi-task learning? In: Proceedings of International Conference on Machine Learning, Venice, Italy, 2020. 9120-9132.
76. Tang H, Liu J, Zhao M, et al. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In: Proceedings of the 14th ACM Conference on Recommender Systems, Online, 2020. 269-278.
77. Jia L, Zhou Z, Xu F, et al. Cost-efficient continuous edge learning for artificial intelligence of things. IEEE Internet of Things Journal, 2021, Vol: 9(10): 7325-7337.
78. Mullapudi R T, Chen S, Zhang K, et al. Online model distillation for efficient video inference. In: Proceedings of the IEEE/CVF International conference on computer vision, California, 2019. 3573-3582.
79. Chen Y, Li W, Sakaridis C, et al. Domain adaptive faster r-cnn for object detection in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Utah, 2018. 3339-3348.
80. Yang C, An Z, Cai L, et al. Mutual contrastive learning for visual representation learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, British Columbia, Canada, 2022. 36(3): 3045-3053.
81. Tsakanikas V, Dagiuklas T. Video surveillance systems-current status and future trends. Computers & Electrical Engineering, 2018, 70: 736-753.
82. Zamir A R, Sax A, Shen W, et al. Taskonomy: Disentangling task transfer learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 3712-3722.
83. Wang Q, Fink O, Van Gool L, et al. Continual test-time domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 7201-7211.
84. Song J, Lee J, Kweon I S, et al. EcoTTA: Memory-Efficient Continual Test-time Adaptation via Self-distilled Regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023: 11920-11929.
85. Zhang J, Tao D. Empowering things with intelligence: a survey of the progress, challenges, and opportunities in artificial intelligence of things. IEEE Internet of Things Journal, 2020, 8(10): 7789-7817.
86. Lin S H, Paolieri M, Chou C F, et al. A model-based approach to streamlining distributed training for asynchronous SGD. In: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE, 2018: 306-318.
87. Niu S, Wu J, Zhang Y, et al. Efficient test-time model adaptation without forgetting. In: International conference on machine learning. PMLR, 2022: 16888-16905.
88. Li T, Huang J, Risinger E, et al. Low-latency speculative inference on distributed multi-modal data streams. In: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services. 2021: 67-80.
89. Dehnavi S, Goswami D, Koedam M, et al. Modeling, implementation, and analysis of XRCE-DDS applications in distributed multi-processor real-time embedded systems. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021: 1148-1151.
90. Gronauer S, Diepold K. Multi-agent deep reinforcement learning: a survey. Artificial Intelligence Review, 2022: 1-49.
91. Park M, Oh H. Cooperative information-driven source search and estimation for multiple agents. Information Fusion, 2020, 54: 72-84.
92. Abad M S H, Ozfatura E, Gunduz D, et al. Hierarchical federated learning across heterogeneous cellular networks. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020: 8866-8870.
93. Zhang X, Zhang A, Sun J, et al. Emp: Edge-assisted multi-vehicle perception. In: Proceedings of the 27th Annual International Conference on Mobile Computing and Networking. 2021: 545-558.
94. Vladimir Vukadinovic, Krzysztof Bakowski, Patrick Marsch, Ian Dexter Garcia, Hua Xu, Michal Sybis, Pawel Sroka, Krzysztof Wesolowski, David Lister, Ilaria Thibault, 3GPP C-V2X and IEEE 802.11p for Vehicle-to-Vehicle communications in highway platooning scenarios,Ad Hoc Networks,Volume 74,2018,Pages 17-29,ISSN 1570-8705
95. Xu, Zhigang , Li, Xiaochi , Zhao, Xiangmo , Zhang, Michael , Wang, Zhongren. (2017). DSRC versus 4G-LTE for Connected Vehicle Applications: A Study on Field Experiments of Vehicular Com-

munication Performance. Journal of advanced transportation. 435. 10.1155/2017/2750452.

96. Akshay Jajoo and Y. Charlie Hu and Xiaojun Lin and Nan Deng.A Case for Task Sampling based Learning for Cluster Job Scheduling. 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). 2022.978-1-939133-27-4
97. Kumar V, Gleyzer L, Kahana A, et al. MYCRUNCHGPT: A LLM ASSISTED FRAMEWORK FOR SCIENTIFIC MACHINE LEARNING. Journal of Machine Learning for Modeling and Computing, 2023, 4(4).
98. Yao S, Hu S, Zhao Y, et al. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In: Proceedings of the 26th international conference on world wide web. 2017: 351-360.
99. Wan S, Qi L, Xu X, et al. Deep learning models for real-time human activity recognition with smartphones. Mobile Networks and Applications, 2020, 25: 743-755.
100. Ma X, Guo F M, Niu W, et al. Pconv: The missing but desirable sparsity in dnn weight pruning for real-time execution on mobile devices. In: Proceedings of the AAAI conference on artificial intelligence. 2020, 34(04): 5117-5124.
101. Guan H, Liu S, Ma X, et al. CoCoPIE: Enabling real-time AI on off-the-shelf mobile devices via compression-compilation co-design. Communications of the ACM, 2021, 64(6): 62-68.
102. https://www.tensorflow.org/lite?hl=zh-cn
103. Jiang X, Wang H, Chen Y, et al. Mnn: A universal and efficient inference engine. Proceedings of Machine Learning and Systems, 2020, 2: 1-13.
104. https://github.com/Tencent/ncnn
105. Wang Z J, Zhan Z H, Yu W J, et al. Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling. IEEE transactions on cybernetics, 2019, 50(6): 2715-2729.
106. Wang L, Yu Z, Yu H, et al. AdaEvo: Edge-Assisted Continuous and Timely DNN Model Evolution for Mobile Devices. IEEE Transactions on Mobile Computing, 2023.
107. Ouyang X, Xie Z, Zhou J, et al. Clusterfl: a similarity-aware federated learning system for human activity recognition. In: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services. 2021: 54-66.
108. Sattler F, Müller K R, Samek W. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. IEEE transactions on neural networks and learning systems, 2020, 32(8): 3710-3722.
109. Boubiche D E, Imran M, Maqsood A, et al. Mobile crowd sensing–taxonomy, applications, challenges, and solutions. Computers in Human Behavior, 2019, 101: 352-370.
110. Li J, Hong D, Gao L, et al. Deep learning in multimodal remote sensing data fusion: A comprehensive review. International Journal of Applied Earth Observation and Geoinformation, 2022, 112: 102926.
111. Gale T, Narayanan D, Young C, et al. MegaBlocks: Efficient Sparse Training with Mixture-of-Experts. Proceedings of Machine Learning and Systems, 2023, 5.
112. Zhao X, Zhou K, Zhang B, et al. JiuZhang 2.0: A Unified Chinese Pre-trained Language Model for Multi-task Mathematical Problem Solving. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2023: 5660-5672.
113. Ausavarungnirun R, Chang K K W, Subramanian L, et al. Staged memory scheduling: Achieving high performance and scalability in heterogeneous systems. ACM SIGARCH Computer Architecture News, 2012, 40(3): 416-427.
114. Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. 2009.
115. Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, et al. 2013. A public domain dataset for human activity recognition using smartphones. In Esann, Vol. 3. 3.
116. Lin J, Zhu L, Chen W M, et al. On-device training under 256kb memory[J]. Advances in Neural Information Processing Systems, 2022, 35: 22941-22954.