ArtFL: Exploiting Data Resolution in Federated Learning for Dynamic Runtime Inference via Multi-Scale Training

Siyang Jiang[†]
The Chinese University of Hong Kong
Hong Kong SAR, China
syjiang@ie.cuhk.edu.hk

Xian Shuai[†]
The Chinese University of Hong Kong
Hong Kong SAR, China
sx018@ie.cuhk.edu.hk

Guoliang Xing[✉]
The Chinese University of Hong Kong
Hong Kong SAR, China
glxing@ie.cuhk.edu.hk

ABSTRACT

Federated Learning (FL) has emerged as a prominent paradigm for distributed machine learning, crucial for mission-critical applications such as autonomous driving and smart health. However, existing FL systems have not adequately addressed the dynamic real-time requirements of these applications due to stringent inference deadlines and resource limitations on edge devices. In this paper, we propose **ArtFL**, a novel federated learning system designed to support dynamic runtime inference through multi-scale training. The key idea of ArtFL is to utilize the data resolution, i.e., frame resolution of videos, as a knob to accommodate dynamic inference latency requirements. Specifically, we initially propose data-utility-based multi-scale training, allowing the trained model to process data of varying resolutions during inference. Subsequently, we introduce an innovative strategy for frame resolution selection in inference, based on the similarity of adjacent frames. Finally, leveraging latency-based dynamic data dropping, we propose a systematic scheme to reduce the overall training time by shortening the waiting time in FL. For evaluation, we build two real-world FL testbeds for smart vehicles and healthcare applications, utilizing a heterogeneous edge platform. Extensive experiments across our testbeds and three public datasets show that ArtFL outperforms state-of-the-art baselines in overall accuracy and system performance up to 36.36% and 47.81%, respectively. A demo video of ArtFL on our smart vehicle testbed is available at https://youtu.be/eeK6yRVEG3U, and our code is available at https://github.com/siyang-jiang/ArtFL.git.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

Federated Learning System, Multi-Scale Training, Real-time System

ACM Reference Format:
Siyang Jiang[†], Xian Shuai[†], and Guoliang Xing[✉]. 2024. ArtFL: Exploiting Data Resolution in Federated Learning for Dynamic Runtime Inference via Multi-Scale Training. In *The ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '24), May 13–16, 2024, Hong Kong*. ACM, New York, NY, USA, 12 pages.

1 INTRODUCTION

Federated Learning (FL) is increasingly adopted in real-time edge applications like autonomous driving [48] and smart health [46]. However, these applications pose two challenges that current FL systems have not addressed. First, during inference, tasks running on edge systems usually impose stringent real-time requirements. A typical real-time deadline for autonomous driving tasks, such as lane-keeping assistance and traffic sign recognition, is ~30 ms (e.g., 33 fps) for each RGB image [61]. Meeting this tight deadline becomes even more challenging due to compute resource contention from other tasks on edge devices. For example, the traffic sign recognition task on a smart vehicle may not immediately access the GPU recourse for inference due to other preemptive high-priority tasks such as passing vehicle detection. The second challenge arises from the system and data heterogeneity that is commonly observed in FL systems. An FL system may consist a large number of clients which may update the network model in an online manner. However, the local training time of these clients may vary significantly due to differences in compute and communication resources, and the amount of available training data. As a result, faster clients may waste time waiting for slower clients, resulting in a prolonged training duration. The advantages of minimizing the wait time can be illustrated by autonomous driving, where updating car aboard models typically needs to be complete with an idle period, such as overnight parking.

Previous efforts in machine learning (ML) systems primarily focused on improving the inference efficiency via model design techniques, such as early exit [12, 21, 52], multi-networks [8, 31, 57], quantization [14, 53] and weight pruning [22, 28]. These techniques are applicable to FL systems for users' privacy preserving. However, these designs either do not adapt to users' different requirements on inference latency and accuracy [28, 53], or increase the storage overhead of the model [12, 57]. Multi-resolution inference [24, 33, 57] is a promising approach to mitigate the above issues, by dynamically adjusting the frame resolution for vision-based tasks during inference. Moreover, considering the training and inference as two closely coupled stage, multi-scale training [11, 24] is shown to be effective in centralized training setting.

However, previous FL systems mainly focused on the training stage without jointly optimizing the training and inference to adapt to users' dynamic inference requirements. In addition, it is not trivial to directly adapt the multi-scale training from centralized learning to federated settings due to a catastrophic performance dropping, as shown in Sec. 2.2. To tackle the system and data heterogeneity commonly present in FL systems, several methods [25, 29, 29, 32, 35, 46, 49, 59] have been proposed. Oort [29], PyramidFL [32], and FedBalancer [49] reduce the training duration via either client or sample selection. In addition, FedProx [35], Scaffold [25], and FedDyn [1] introduce additional loss terms as regularization to align the heterogeneous distribution across clients.

[†] denotes equal contribution, [✉] corresponding author.

979-8-3503-6201-5/24/$31.00 ©2024 IEEE
DOI 10.1109/IPSN61024.2024.00007

However, these methods are not designed to minimize the waiting time, which can be a substantial part of the total training duration.

In this paper, we propose **ArtFL**, a novel FL framework that facilitates dynamic runtime inference through multi-scale federated training. The key idea driven by the understanding that the data resolution can be exploited to achieve desirable trade-offs between learning latency and accuracy. In particular, we first propose *data-utility-based multi-scale training*, which leverages the data utility from clients to guide the selection of training sample resolution. It allows clients to learn scale-invariant features, which are then effectively aggregated by the server. Next, upon model convergence, an *optimal frame resolution selection* scheme is adopted to choose the suitable resolution for each frame, aligning with the dynamic real-time requirements based on the similarity of adjacent frames.. Lastly, to mitigate the increasing of waiting time, we propose a systematic scheme, named *latency-based dynamic data dropping*, which determines the amount of data to be dropped in subsequent training rounds based on empirical training time.

To evaluate ArtFL, we build two testbeds, including an F1/10 smart vehicle testbed and a smart health testbed for patient activity monitoring using heterogeneous devices, including Nvidia Jetson Orin, AGX Xavier, Xavier NX, TX2, and Nano. In addition, we also evaluate ArtFL on a GPU server involving three public RGB image datasets (CIFAR10, CIFAR100, Tiny-ImageNet). Extensive experiment results show that ArtFL significantly outperforms several existing federated learning paradigms in accuracy and incurs less training overhead under dynamic real-time requirements. In particular, ArtFL largely outperforms other existing systems and improves the overall accuracy accuracy and system performance up to 36.36% and 47.81%, respectively.

2 APPLICATION AND MOTIVATION

2.1 Application Scenarios

ArtFL can be applied to various distributed systems where each client consists of a camera and an edge compute unit and collaborates with other clients to learn a deep learning model while preserving the raw data privacy and reducing the training expense. We now use autonomous driving as a representative example to discuss the application of ArtFL [61]. For instance, a group of smart vehicles can form an FL system that learns the unseen knowledge from other clients and avoids the overfitting issue while each client only uses local data for training [44]. Due to the time-critical nature and the high compute overhead of autonomous driving tasks, existing autonomous cars such as Tesla [9] or XPeng [47] have to be equipped with several high-end edge devices, i.e., NVIDIA Jetson Orin [45]. For example, serving a real-time object detection task with RetinaNet [38] in single stream mode needs 8.19 FP32 TFLOPS [45], which poses a heavy workload for even high-end devices like Orin. Another major challenge is that during inference, the computing resources available for a real-time task are typically dynamic due to other preemptive tasks with higher priority. For example, the typical traffic sign recognition task deadline is ~30 ms (e.g., 33 fps) of each inference. Higher-priority detection tasks, such as collision detection, may preempt or interrupt the execution of lower-priority tasks, such as traffic sign recognition. To handle such dynamics, a task should be able to adjust its execution delay at runtime. As another example, in the presence of an imminent safety threat, an autonomous driving system may increase the sampling rate of onboard sensors while requiring the same level of accuracy, resulting in a tighter deadline for learning tasks. However, achieving such latency-accuracy adaptivity is highly challenging since the current approaches train deep models with a constant level of accuracy by fixing model parameters and structure, leading to a fixed inference latency. ArtFL aims to address the above challenges by adaptively tuning the frame resolution during the inference to meet the dynamic real-time requirement. Moreover, thanks to the specifically designed FL training scheme, ArtFL not only achieves the adaptive frame resolution tuning during inference with minimal accuracy loss but also provides high FL training efficiency.

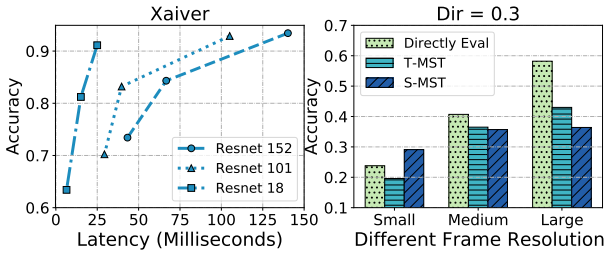


Figure 1: A motivation study. *Left*: A range of the accuracy and inference latency trade-offs. *Right*: The result of adopting direct evaluation and two vanilla multi-scale approaches, e.g., temporal and spatial multi-scale training.

2.2 A Motivation Study

Trade-off between the accuracy and latency. Reducing input resolution is one of the most effective methods for visual tasks to enhance inference efficiency [24]. Shown in the left part of Fig. 1, we train and evaluate three different depth neural networks under different scale resolutions (32×32, 16×16, 8×8). This experiment observed that using ResNet 101 and the original size for inference presented a considerable challenge for NVIDIA Jetson Xavier in achieving an inference latency lower than 100ms, i.e., a frame rate of >10 fps. However, using medium and small frame resolutions effectively achieved the required real-time performance. In particular, the smaller resolution reduces inference time from 100ms to 30ms, with an accuracy reduction from 92% to 70%. In summary, the data resolution provides models significant room for the trade-off between accuracy and real-time performance.

Effect of training and inference resolution discrepancy. As discussed above, the reduction of input resolution has been identified as an effective means of achieving a considerable speedup while minimizing accuracy degradation. Consequently, prior works [21, 60] have employed dynamic resolution tuning during inference. However, we show that directly scaling original data into a medium or small resolution during inference time is not viable in federated learning. In particular, we trained a neural network using high-resolution data and directly evaluated its performance among three resolutions. As shown in *(Directly Eval Bar)* of Fig. 1 (right part), we observed a reduction in accuracy of up to 30% when using small resolution in inference, with Dir=0.3 which is a widely used parameter to control the data non-i.i.d. level in federated learning [34].

This result is commonly referred to as the training and testing resolution discrepancy, as discussed in [54]. During training, the model acquires features under specific resolutions, often poorly transferable to different resolutions employed during inference. Therefore, a unified framework that can effectively handle such different resolutions for inference is necessary.

Challenges of multi-scale training in FL. Multi-scale training has been shown effective in centralized training [24, 33, 57]. However, adopting a multi-scale technique in federated learning is not trivial. For example, we integrate two vanilla multi-scale techniques from both spatial and temporal perspectives. In the temporal multi-scale training (T-MST), we use three different resolutions, i.e., small, medium, and large, and one of these resolutions is sequentially chosen to train the neural network after an aggregation. For spatial multi-scale training (S-MST), we assigned different frame resolutions to separate clients during the training process. Specifically, we have partitioned the clients into three equal groups with small, medium, and large resolutions. As shown in Fig. 1 (right part), our empirical evaluation reveals that both methods suffer from substantial accuracy degradation on high resolutions. This result illustrates the challenge of directly incorporating multi-scale training into FL. The phenomenon we are observing can be primarily attributed to a well-acknowledged limitation in neural networks known as catastrophic forgetting. In particular, within the context of the T-MST, the intricacies of this issue come to the forefront. As the network is exposed to input data at varying resolutions over time, the newly acquired features at higher resolutions tend to overwrite the foundational knowledge that the network initially learned from the lower resolutions. This results in the erosion of previously established patterns and a degradation in the network's ability to recall earlier learned information. Similarly, in the S-MST scenario, a parallel but distinct form of forgetfulness manifests. When multiple clients are involved, each client's neural network specializes in processing data at specific resolutions. However, when these networks are required to integrate and generalize across different resolutions handled by other clients, they often fail to retain the feature representations learned from those disparate data scales. This leads to a scenario where the information pertinent to one client's resolution is forgotten when the network attempts to accommodate and learn from the resolutions of others.

In addition, the waiting time is generally prolonged after adopting multi-scale training due to the disparity in the training data volume and computational resources. It means that faster clients waste more time waiting for slower ones. In particular, we deployed the spatial federated learning system on a GPU server and an edge platform comprising ten heterogeneous devices. We calculated the transmitting time, training time, and waiting time separately. In Fig. 2 (Left Part), waiting time and training time account for nearly equal proportions, constituting the majority of the overall process duration within the edge devices platform. However, our observations indicate that multi-scale training further exacerbates this situation, as shown in Fig. 2 (right part). In particular, we observed a significant increase in the average waiting time, which rose from 39% to 46% and 46% to 60% upon adopting the multi-scale training approach in our GPU and edge platform (demonstrated by red arrow), respectively.

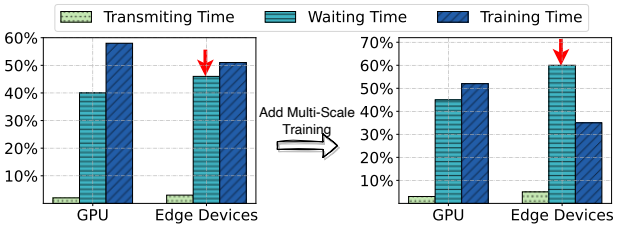


Figure 2: Adopting the spatial multi-scale training, the percentage of waiting time increased in GPU and edge devices.

2.3 Summary

To meet the dynamic requirements during inference, our motivation study underscores the potential that data resolution offers to models, enabling a significant trade-off between accuracy and latency. However, it was observed that direct inference on different data resolutions under conventional training can drastically undermine accuracy due to resolution discrepancy between training and inference. This observation motivated the exploration of multi-scale training, treating training and inference as two closely coupled stages. Nevertheless, several challenges need to be addressed for integrating multi-scale training into FL due to accuracy drop and potential training inefficiency.

3 PROBLEM FORMULATION AND OVERVIEW

3.1 Problem Formulation

Without loss of generality, we assume there are N clients involved in a FL system, and the training data owned by client i is $\mathcal{D}^i_{train}$, where $i \in \{1, 2, ..., N\}$. Consistent with prevalent data heterogeneity, the distribution of $\mathcal{D}^i_{train}$ differs among clients. Moreover, the compute capability C^i_{comp} and the communication bandwidth C^i_{comm} of each client may be distinct. We use $\mathbf{r} = \{r^j\}_{j=1}^{J}$ to denote the set of J different frame resolutions, θ to denote the obtained model weight from federated training. Given an inference latency requirement T, our objective is to obtain a robust model θ, which can achieve accuracy as high as possible for each client by choosing a proper resolution r^j for input samples while minimizing the training time. Note that during inference, the latency requirement of a given task is dynamic due to various reasons, such as unpredictable system delays and resource contention from other preemptive tasks with higher priority. We assume that the data of the highest resolution that neural network models need can be stored in all devices since, in practice, the resolution of cameras (e.g., 720p, 1080p) usually meets (and even exceeds) the resolution required by the neural network model (e.g., 64×64 or 224×224).

3.2 System Overview

The goal of ArtFL is to exploit the frame resolution as a knob to meet the requirement of real-time performance and accuracy in the dynamic inference stage while keeping the training phase as compute-efficient as possible. The core ideas of ArtFL include *multi-scale training*, which enables flexibility in the inference, and *frame resolution selection* ensures real-time performance, which enhances the accuracy performance during the inference. Meanwhile, we

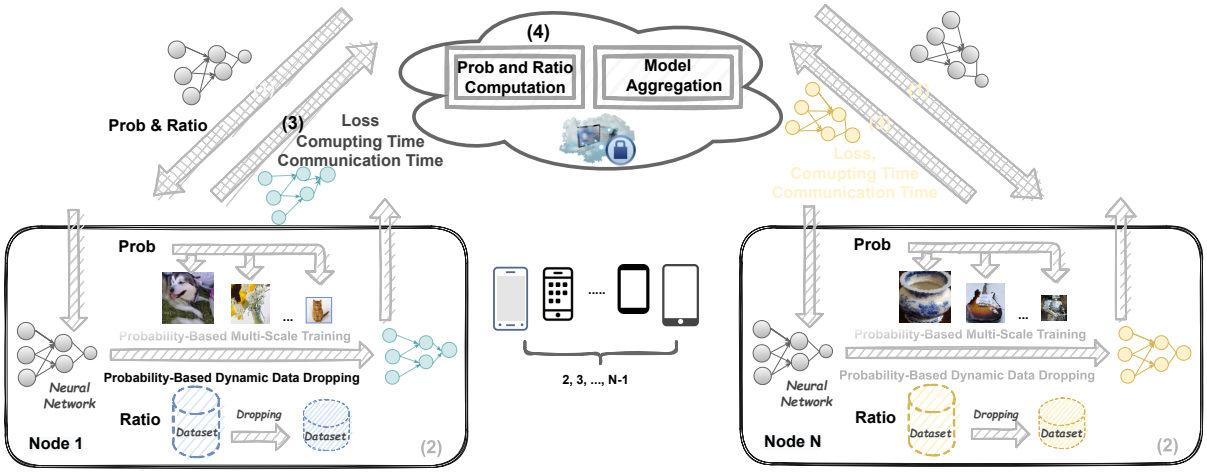


Figure 3: Pipeline of ArtFL. The server aggregates clients' models and computes the probability vector for different resolutions and data dropping ratios for each client. The clients execute the local update, guided by the probability vector and the ratio.

propose *dynamic data dropping* in the training stage to decrease the waiting time.

The overall system pipeline of ArtFL is shown in Fig. 3. First, clients check in to the server and then receive the global model and the metadata from the server, including the probabilities on different resolutions and the dropping ratio for training data. Next, guided by the metadata, clients locally update the received global model using multi-scale training with the clipped dataset. Then, once the local training is finished, clients upload the well-trained model and report the training time consumption and loss. Last, the server aggregates the model and calculates the metadata for each client for the following FL round. Each communication round consists of the above steps until convergence.

We give a brief illustration of multi-scale training and dynamic data dropping. For multi-scale training, we sample different resolutions following a certain probability vector on every client inspired by a combination of temporal and spatial multi-scale training mentioned in Sec. 2.2. This enables each client to extract the scale-invariance knowledge and let the server easily aggregate such knowledge. Notably, the acquisition of the probability vectors is based on the training loss of each client, which is an indicator of local data utility [32]. Moreover, to avoid the long waiting time for stragglers and to speed up the federated training, slow clients will drop some data. The data dropping ratios for all clients are jointly coordinated by the server on the basis of their training time prediction. The details are in Sec. 4.1. In the inference, the core idea of optimal frame resolution selection is to select the resolution with similarity between two adjacent frames for each frame. We realized that only the keyframe (with a small similarity) should be assigned the most significant resolution. Note that in ArtFL, the training and inference processes are not optimized jointly. In other words, the techniques applied in the training and inference stages are independent of each other. We mainly consider the training and inference together since we aim to introduce dynamic optimization to the inference stage. To accomplish this, we employ a data-utility-based multi-scale training approach to link the two stages.

4 DESIGN OF ARTFL

4.1 Data-Utility-Based Multi-Scale Training

To enable the flexibility of dynamic requirements in inference time, we consider the training and inference as two closely coupled stages and then propose data-utility-based multi-scale training.

As mentioned in Sec. 3.1, there are J possible frame resolutions for the data on each client. For each client i, we use a vector $\mathbf{p}^i \in \mathbb{R}^J$ to denote the probability of selecting every resolution during the local training. In order to obtain the suitable probability vector $\mathbf{p}^i$, we collect the average per-sample training loss L^i for each client:

$$L^i = \frac{1}{|\mathcal{D}^i_{train}|} \sum_{(x,y)\in\mathcal{D}^i_{train}} loss(x,y) \tag{1}$$

Intuitively, L^i indicates the difficulty level of the data on client i, where clients with larger loss may include more hard examples and thus should be assigned with a higher probability of the large resolution to extract more detailed features for model improvement. To this end, we let the server collect L^i from all clients to form $L = \{L^i\}_{i=1}^N$. We then sort L and obtain the ranking of each client. We can assign each client a $\mathbf{p}^i$ from a pre-defined ordered pool $\mathbf{Q}$ based on the ranking. The higher the loss, the vector with a higher probability towards a large resolution will be assigned. We note that L^i can be easily logged during training, and thus, the acquisition of $\{\mathbf{p}_i\}_{i=1}^N$ incurs negligible overhead.

In practice, instead of using a pool with a size that equals to the number of clients, we can construct the pool $\mathbf{Q}$ with only S levels of probabilities, i.e., $\mathbf{Q} = \{\mathbf{q}_s\}_{s=1}^S$, where $\mathbf{q}_s \in \mathbb{R}^J$ and $S < N$. For simplicity, the probability vectors of N clients can be equally divided into S preference levels based on the loss ranking, and several clients within the same level may share the same probability vector. For instance, consider the case shown in the upper part of Fig. 4, where $J = 3$ and $S = 3$ (i.e., three resolutions and three preference levels). A typical pool is $\mathbf{Q} = [(0.2, 0.2, 0.6), (1/3, 1/3, 1/3), (0.6, 0.2, 0.2)]$, including three probability vectors. The first vector of $\mathbf{Q}$ is (0.2,

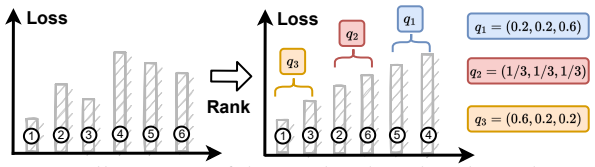


Figure 4: Illustration of data-utility-based multi-scale training and numbers indicate the client indices.

0.2, 0.6), refers to a probability of 0.6 to choose the large resolution while 0.2 to choose the small and medium resolution during the training and is assigned to clients with the loss belonging to the top one third (high). Note that the number of selectable resolutions can increase, such as offering 5 or 10 distinct options of $\mathbf{Q}$.

Note that our design is based on several insights. First, we let the samples with different resolutions appear alternately during local training to ensure that each client gradually learns scale-invariant knowledge. Second, we design a global probability vector assignment scheme that allows the clients that currently accumulate a larger loss to be paid more attention (i.e., assigned with a higher probability on large resolution) in future training. In addition, compared with the cross-device multi-scale training approach introduced in Sec. 2.2, our design alleviates the " non-i.i.d. size issue". It allows the learned scale-invariant features by each client to effectively be aggregated among a series of FL rounds. It is worth noting that in ArtFL obviates the necessity for storing multiple resolutions about the same objects.

4.2 Optimal Frame Resolution Selection

In Sec. 4.1, we enable desirable flexibility to model for achieving dynamic real-time requirements during inference. In the following, we propose optimal frame resolution selection to choose the optimal resolution for each frame according to the similarity of two adjacent frames.

First, for any client, we need to obtain the optimal strategy $S = \{s_i\}_1^J$, where s_i represents the probability of choosing different resolutions under a dynamic time requirement T. For example, the probability of choosing the largest resolution (i.e., s_J) under a tight time budget could be small. In contrast, the probability of choosing the smallest resolution (i.e., s_1) might be relatively large. In the following, we formulate this problem into a linear optimization:

$$\max_{s_i \in S} \sum_i^J a_i s_i, \quad s.t. \begin{cases} \sum_i^J t_i s_i \le T \\ \sum_i^J s_i = 1 \end{cases}$$

where, t_i denotes the inference time of i^{th} resolution, and a_i is the accuracy of i^{th} resolution in the training set. To solve this optimization problem, we utilize the simplex method [43] with an average time complexity of $O((\mathcal{N} + \mathcal{M}) * \mathcal{N})$. Here, $\mathcal{N}$ represents the number of variables, and $\mathcal{M}$ denotes the number of inequality constraints. The search space is limited in practice since $\mathcal{N}$ and $\mathcal{M}$ are small.

Second, with the strategy S, we need to determine the specific resolution for each frame. To this end, we use the structural similarity index measure (SSIM) metric [55] for adjacent two frames x_k and x_{k+1}. Let $\hat{p} \in \mathbb{R}^J$ denote the probability of selecting resolution in frame x_k, we have

$$\hat{p} = \begin{cases} [0, \cdots, 1] \in \mathbb{R}^J & , \text{SSIM}(x_k, x_{k+1}) \le \beta \\ S & , else, \end{cases} \tag{2}$$

where β is a hyperparameter defined in advance. We use $\beta = 0.3$ in this paper. As shown in Eq. 2, we use the highest resolution when the two adjacent frames have a huge difference. For example, suppose that $J = 3$ and $\text{SSIM}(x_k, x_{k+1}) \le \beta$, $\hat{p} = [0, 0, 1]$. To satisfy the time constraint, we need to update the optimal strategy S, shown as follows. For each element $s_i \in S$, $S \in \mathbb{R}^J$, we have

$$s_i = \begin{cases} (s_i - T)/(1 - T), & i = K, s_i \ge T \\ s_i/(1 - T), & else, \end{cases} \tag{3}$$

where K is the index of the latest selected resolution. The core idea of optimal frame resolution selection is that SSIM presents the changes between keyframes needing higher resolutions.

4.3 Latency-Based Dynamic Data Dropping

Model training efficiency plays a critical role in FL due to the online nature of training and regular model updates in applications [4, 42]. As demonstrated in Sec. 2, the waiting time is prolonged in the training stage due to the multi-scale training. This observation motivated us to propose *latency-based dynamic data dropping* aims to reduce the training duration.

4.3.1 Training time estimation. The local training time of each client is mainly influenced by the number of samples and the taken resolution probabilities. Instead of a learning-based model, we accomplish the latency prediction using an intuitive approach that utilizes the measured latency of the last round as a reference. Our insight is that the last round of training time can be obtained without any extra effort, and based on which, the latency of the next round can be easily inferred after the transformation:

$$T_{E\_Train}^{(i,t)} = \frac{\mathbf{p}^{(i,t)} \otimes \mathbf{I}}{\mathbf{p}^{(i,t-1)} \otimes \mathbf{I}} \cdot \frac{1}{Dr^{(i,t-1)}} \cdot T_{Train}^{(i,t-1)}, \tag{4}$$

where the superscript i and t refer to the index of the client and the global FL round, respectively. $T_{E\_Train}^{(i,t)}$ and $T_{Train}^{(i,t-1)}$ denotes the estimated training time and measured training time of the i^{th} client in round (t) and $(t-1)$, respectively. $\mathbf{p}^{(i,t)}$ denotes the probability vector, and $\mathbf{I} = \mathbf{r}^2 \in \mathbb{R}^J$ denotes the compute intensity level of using different resolutions[1], where $\mathbf{r}$ is defined in Sec. 3.1. The dot production of the probability vector and the compute intensity represent the computational overhead. $Dr^{(i,t-1)}$ the corresponding data dropping ratio, which will be defined in Eq. (6), and $\otimes$ the dot product operation. Notably, the obtained time estimation $T_{E\_Train}^{(i,t)}$ is under the assumption that the full local dataset on client i is used in the next round. Utilizing this estimation, the server can dynamically adjust the data dropping ratio for clients to prevent prolonged local updates, reducing the waiting time.

[1]Here, the quadratic relationship is because the compute overhead of convolutional layer is quadratic to the image size.

4.3.2 Dynamic data clipping. As the computing power and the number of possessed data on the client vary a lot, the client that first finishes the local training may have to wait for the last one for a while, and our goal is to minimize waiting time as much as possible while preserving performance.

Thus, we propose dynamic data clipping. Each client must discard some data, with longer training duration mandating more deletions. Our insight is that as the majority of clients have completed the local training, the averaged model update will not be substantially changed by a few stragglers' model updates, and dynamic data dropping will not significantly influence the model accuracy.

However, the networking situation is also heterogeneous across mobile devices, leading us to consider the communication time in the training phase. In particular, we estimate the transmitting time as follows:

$$T_{E\_Comm}^{(i,t)} = \gamma T_{E\_Comm}^{(i,t-1)} + (1-\gamma) T_{Comm}^{(i,t-1)} \tag{5}$$

where, $T_{E\_Comm}^{(i,t-1)}$ and $T_{Comm}^{(i,t-1)}$ denote the estimated communication time and the real one in round $(t-1)$ of the i^{th} client, respectively. γ is a hyper-parameter controlling the momentum effect of previous communication time, and we use $\gamma = 0.01$ in this work.

In the following, we use a dropping function Dr to determine how much percentage of data should be dropped:

$$Dr^{(i,t)} = \begin{cases} (T_E^{(i,t)} - \overline{T}^t)/T_E^{(i,t)} & , T_E^{(i,t)} > \overline{T}^t \\ \alpha & , T_E^{(i,t)} < \overline{T}^t \end{cases} \tag{6}$$

where $T_E^{(i,t)} = T_{E\_Comm}^{(i,t)} + T_{E\_Train}^{(i,t)}$. It denotes the predicted time of client i in round t^{th} round. $\overline{T}^t$ denotes the average estimated time among all clients. α is the basic dropping rate, where we select $\alpha = 0.1$ in this work. As can be seen, we let clients whose predicted training time is above the average perform the data clipping, and force their new expected time to be no longer than the original average. Notably, this component plays a pivotal role in enhancing the efficiency of the federated learning training process, particularly when some clients have substantially lower computing capabilities than others.

Note that in ArtFL a high-loss client is more likely to obtain a higher resolution in a round, but this does not imply that it will drop more data in the next round since the server determines the dropped ratio on each client by considering both the per-client data utility and running efficiency. The worst-case scenario involves clients dropping a significant amount of data, which can lead to difficulties in achieving convergence. However, ArtFL can mitigate this issue by leveraging knowledge from other clients. In particular, the server aggregates the knowledge distilled from all participating clients. This aggregation process has the potential to compensate for information that may be lost due to data omission on clients. The weights shared across the network encapsulate features within the semantic space, effectively representing the collective knowledge acquired during the training process.

4.4 Put It All Together

We now summarize ArtFL which is presented in Algorithm 1.

Algorithm 1 ArtFL

Require: Local dataset $\mathcal{D}_{train}^i$; Number of Clients N

1: **for** each round t **do**
2: Sample clients $P^t \subseteq [N]$
3: Obtain $\mathbf{p}^{(i,t)}$, for all $i \in P^t$ ▷ Sec. 4.1
4: Predict $T_{E\_Train}^{(i,t)}$, for all $i \in P^t$ ▷ Sec. 4.3.1
5: Predict $T_{E\_Comm}^{(i,t)}$, $T_E^{(i,t)}$ for all $i \in P^t$ ▷ Sec. 4.3.2
6: Obtain $Dr^{(i,t)}$, for all $i \in P^t$ ▷ Sec. 4.3.2
7: Server transmits θ, $\mathbf{p}^i$, $Dr^{(i,t)}$ to selected clients
8: **for** each client $i \in P^t$, and in parallel **do**
9: Random clip $\mathcal{D}_{train}^i$ with ratio $Dr^{(i,t)}$ for $\hat{\mathcal{D}}_{train}^i$
10: Multi-scale training with $\mathbf{p}^i$ on dataset $\hat{\mathcal{D}}_{train}^i$
11: Upload the updated model $\hat{\theta}^i$, the measured training time $T_{Train}^{(i,t)}$, the communication time $T_{Comm}^{(i,t)}$, and the loss L^i to the server
12: Model Aggregation: $\theta = \sum_{i \in P^t} \frac{|\hat{\mathcal{D}}_{train}^i|}{\sum_i |\hat{\mathcal{D}}_{train}^i|} \hat{\theta}^i$
13: Adopting optimal frame resolution selection with different resolutions in inference. ▷ Sec. 4.2

(1) Line 3: On the server, acquisition of $\mathbf{p}^i$ is based on the ranking-based scheme introduced in Sec. 4.1.
(2) Line 4-6: Using $\mathbf{p}^i$, the server estimates the local training time $T_E^{(i,t)}$ for each client assuming the full local dataset was used, and then estimates the commutation time $T_E^{(i,t)}$, and in the next calculates $Dr^{(i,t+1)}$.
(3) Line 7: Model and metadata are downloaded from the server to the client.
(4) Line 8 - Line 12: Every selected client performs probabilistic multi-scale training based on the vector $\mathbf{p}^i$ using the clipped datasets. Lastly, the updated model will be sent to the server for aggregation, similar to the conventional federated learning approach.
(5) Line 13: Once obtain the aggregated model θ, optimal frame resolution selection is adopted in inference stage.

5 EXPERIMENTS SETUP

5.1 Baselines

We identified two groups of baseline methods for comparison. The first group pertains to FL and encompasses FedAvg [41], FedProx [35], FedDyn [1], and BalanceFL [50]. The second group focuses on on-device inference systems like FlexDNN [12] and RANet [57], explained in Sec. 6.2. The main reason for selecting FlexDNN and RANet is their dynamic behavior during inference. In contrast, other state-of-the-art (SOTA) on-device inference techniques, e.g., quantization or pruning, only speed up the inference time without considering the dynamics in our setting.

- **FedAvg [41]:** the standard FL approach, where all clients use the conventional cross-entropy loss for training.
- **FedProx [35]:** an FL approach that tackles the heterogeneity issues among clients. Compared with FedAvg, a *L2* regularization term is added to restrict the distance between the local model and the global model.

- **FedDyn [1]:** another state-of-the-art FL solution to the heterogeneity issue. It uses a dynamic regularizer for each device at each round, so that the optimal model for the regularized loss is in conformity with the global empirical loss.
- **BalanceFL [50]:** A state-of-the-art FL framework tackles the data imbalance heterogeneity issue among clients. We include BalanceFL as one of the baselines to investigate whether its knowledge inheritance technique could also be effective.
- **Local Training:** the model trained using only the local data at each node, with a traditional cross entropy loss.
- **Centralized Training without Multi-Scale Training (CT W. O. MST):** the model trained on an overall dataset aggregating all distributed data, using the traditional cross-entropy loss. Note that our experiments are under the generic FL setting.
- **Centralized Training with Multi-Scale Training (CT W. MST):** Compared to CT W. O. MST, the difference is that adding the multi-scale training [16] in centralized training, which can be regarded as the upper bound of our setting.

5.2 Implementation

We implement a prototype of ArtFL on a real edge device and a cloud server platform. Notably, we use the edge platform to evaluate three real-world testbeds while using the more powerful cloud server to evaluate three public datasets for system scalability. Table 1 summarizes the setting. The cloud server is equipped with Intel(R) Core(TM) i9-9820X CPU and 4 NVIDIA A100 GPUs. On the real edge device platform, we use a personal computer (only used for aggregation) as the server and NVIDIA Jetson Orin, AGX Xavier, Xavier NX, TX2, and Nano as clients. The server and nodes are connected within an intranet via a TP-LINK Switch (TL-SG2016K), as shown in Fig. 5. Note that we only present the partial key components of our hardware testbed for simplicity. For the cloud server, we create one process for each client to simulate multiple clients in federated learning and let another process serve as a server, which executes the node selection and model aggregation. We summarize the settings of these two platforms in Table 1. We create one process for each client to complete the local training using GPU.

The deep learning component of our implementation is based on PyTorch. For most tasks, we utilize ResNet-8 [50], which consists of 7 convolutional blocks and a single fully connected layer, while we employ ResNet-18 [50] in the smart health testbed,. It is worth noting that we incorporate an adaptive pooling layer to accommodate various resolutions in the input data. We set the maximum global round to 200 for all federated learning methods with five local epochs each round. SGD optimizes local updates with a learning rate of 0.005 and a momentum of 0.9. The local batch size is 64, and we present the average results of the final five epochs in each baseline. For FedProx, the hyper-parameter μ to control the regularization strength is set to 0.05.

6 REAL-WORLD TESTBED EVALUATION

6.1 Two Real-world FL Testbeds

Here, We present our smart vehicle and smart health FL testbeds.[2]

[2]The data collection was approved by IRB of the authors' institution.

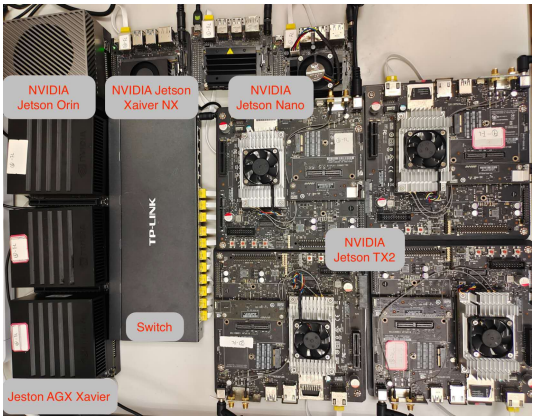
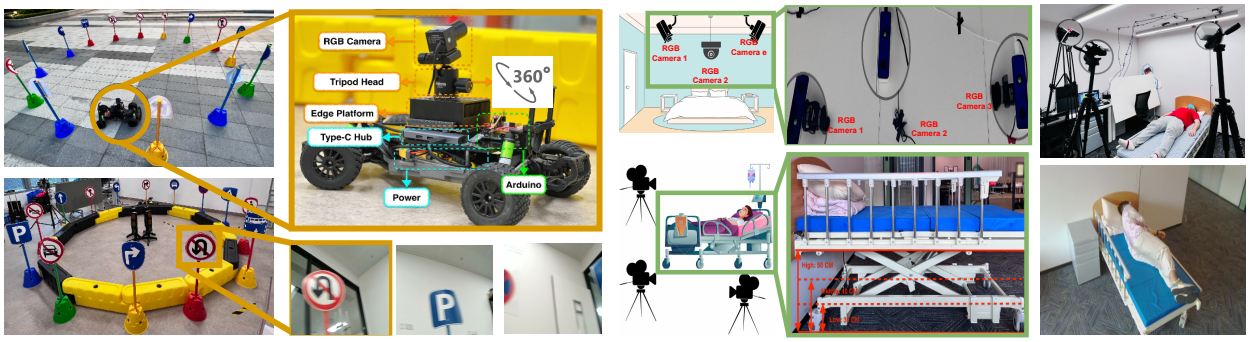


Figure 5: Hardware Setup.

Table 1: A summary of two platforms.

Platform	Details
Edge Device Platform	Jetson Orin * 1. Cores of CPU/GPU: 12/1792. DRAM: 32G
	AGX Xavier * 3. Cores of CPU/GPU: 8/512, DRAM: 16G
	Xaiver NX * 18. Cores of CPU/GPU: 6/384. DRAM: 8G.
	Jetson TX2 * 6. Cores of CPU/GPU: 6/256. DRAM: 8G.
	Jetson Nano * 2. Cores of CPU/GPU: 4/128. DRAM: 4G.
Cloud Sever	CPU: i9-9820X. GPU: NVIDIA A100 * 4. DRAM: 512G.

- **Smart Vehicle Testbed.** In this testbed, we simulate an autonomous driving application where the task is to recognize the traffic signs in real time ($\geq$ 30 fps). As shown in Fig 6(a), the F1/10 Autonomous Vehicle [50] is equipped with a camera, Arduino, and an embedded platform (NVIDIA Xavier). This testbed comprises 15 classes, including "No Driving," "No Right Turn", "No U-Turn", "No Entry", "No Parking", "Vehicle Driving", "Vehicle Parking", "Keep Straight", "Bus Lane", "Turn Left", "Turn Right", "Pedestrian Crossing", "Unknown". In total, we collected 25,593 RGB images with a resolution of 640×480 from different camera angles (left, right), camera heights (high, low, medium), different backgrounds (indoor and outdoor), and light intensity (strong or weak). We use the Dirichlet function to assign these samples to 24 smart cars (clients).
- **Smart Health Testbed.** This testbed simulates the scenarios of patient monitoring of two places: the hospital (sick room) and home (bedroom), where the key difference between the two places is the camera's positions and views, as shown in Fig. 6. We expect the system to detect patients' calls for assistance in time (5 fps) as long as patients make a simple gesture such as raising their right hand. This function can detect the patient's cerebral thrombosis if they are not sleeping supine to avoid exacerbating poor blood flow. In each room, the bed has three different heights (high, medium, and low). Three RGB cameras (Vzense DCAM710) are placed at three different angles (left, middle, and right). During the data collection, we recruited 21 volunteers, and 31,010 RGB images with a resolution of 480×360 were collected. The system is trained to recognize eight different actions: "raising the left/right hands", "lying on the bed", "sitting on the bed", "sitting on the bedside", "lying on the left/right side", and "standing on



(a) Smart Vehicle Testbed.

(b) Smart Health Testbed.

Figure 6: Two Real-world FL Testbeds.

the bedside". During the training, each human identity forms a separate FL client.

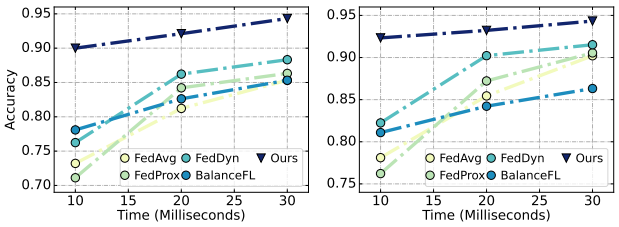
6.2 Accuracy Comparisons under Time Bounds

Comparison with FL systems. We evaluate the accuracy performance under different inference latency constraints of ArtFL and other FL baselines. For the smart vehicle testbed, we define two different levels of compute overloads. In spare time, only a ResNet-8 model is running for traffic sign recognition, while in peak time, another pre-loaded Tiny-YOLO-v3 model is running as a background task. Under both settings, we set the real-time deadline as 10, 20, 30ms (≥ 33 fps). We set the real-time deadline for the smart health testbed as 0.14s, 0.18s, and 0.20s (≥ 5 fps). The main reason why we use different time constraints since we want to simulate the dynamics in real applications. Similarly, there are two real-time deep learning tasks, e.g., patient action recognition and a background task for human falling detection. The results in Fig. 7 show consistent patterns with both testbeds. Note that the x-axis of Fig. 7 refers to different task deadlines. In particular, we observe that ArtFL performs well in both peak and spare time. When the time bound is tight, baseline performance degrades considerably, while ArtFL maintains relatively consistent performance thanks to the multi-scale training.

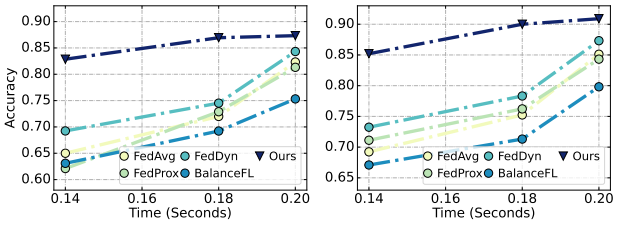
Comparison with input-adaptive systems. We also compare ArtFL with the following two adaptive inference frameworks in the FL setting where the training is completed with FedAvg. We follow the original hyperparameters settings in FlexDNN [12] and RANet [57].

- **FlexDNN [12]:** an input adaptive DNN-based framework for efficient inference with an early exit scheme. Note that in FlexDNN, the input resolution is not changed, and early exit only depends on the difficulty of the input image.
- **RANet [57]:** A multiple models (branches) framework for a faster inference by adapting different resolution representations. Low resolution is used for classifying "easy" inputs while "hard" samples are used for high resolution.

As shown in Table 2, ArtFL outperforms both baselines by a large margin under the time constraints. In particular, ArtFL achieves above 94% and 85% accuracy in intelligent vehicles and health testbeds, while the other two methods can only achieve less than 60%, which is not practical in our natural systems. The main reason



(a) Vehicle testbed on NVIDIA Jetson Xavier.



(b) Health testbed on NVIDIA Jetson Nano.

Figure 7: *Left* and *Right* part presents the accuracy comparison in *Peak* and *Spare* time, respectively.

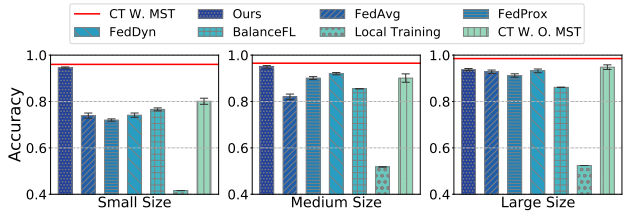
Table 2: Accuracy Comparison under Real-Time Constraint.

Testbeds	Runtime Status	FlexDNN[12]	RANet[57]	Ours
Smart Vehicle	Peak	0.43	0.44	**0.93**
Smart Health		0.39	0.35	**0.81**
Smart Vehicle	Spare	0.58	0.51	**0.95**
Smart Health		0.47	0.43	**0.87**

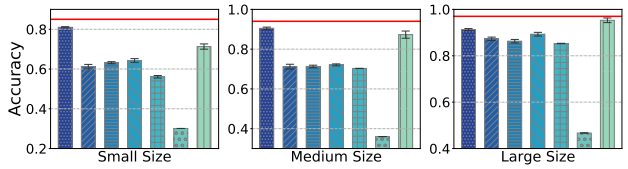
is that the data in the real world is not as clean as in public datasets, with many hard examples, which is not friendly to this early exit scheme and multi-branches-based method.

6.3 Comprehensive Analysis of Accuracy and System Performance

Here, we present the detailed accuracy of various baselines under different inference sizes, shown in Fig. 8. We observe that all approaches tend to have a higher recognition accuracy in the large size except local training due to the lack of multi-client cooperation among three testbeds. Second, ArtFL consistently outperforms other



(a) Smart Vehicle Testbed.



(b) Smart Health Testbed.

Figure 8: Accuracy on two Real-World Testbeds.

Table 3: Average Waiting Time (Seconds) and Mean Local Update Time (Seconds) on three FL Testbeds.

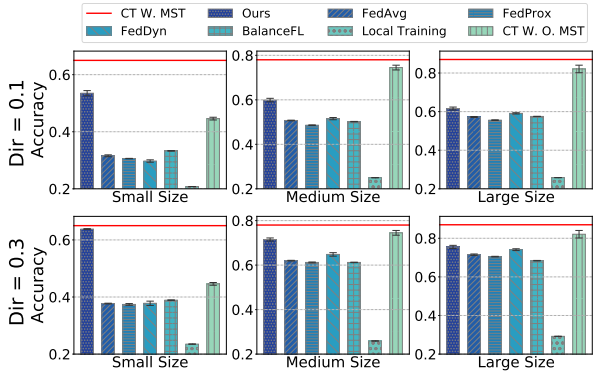
Testbed	FedAvg	FedProx	FedDyn	BalanceFL	Ours
Average Waiting Time					
Smart vehicle	156.43	164.43	169.51	211.62	**110.92**
Smart Health	226.51	231.10	234.88	301.61	**131.79**
Mean Local Update Time					
Smart vehicle	134.52	139.21	141.23	197.83	**101.21**
Smart Health	351.23	376.41	371.96	534.24	**227.35**

baselines in various settings. For example, in small size, ArtFL relatively improves the accuracy with outperforms by 40.38%, 45.29%, 67.35%, 50.13%, and 71.20% over FedAvg, FedProx, FedDyn, BalanceFL and Local training in smart vehicle testbed, respectively. It also shows similar trends in smart health, showing the generality of ArtFL. In addition, we observe that only the performance of ArtFL can achieve over 90% in the middle size.

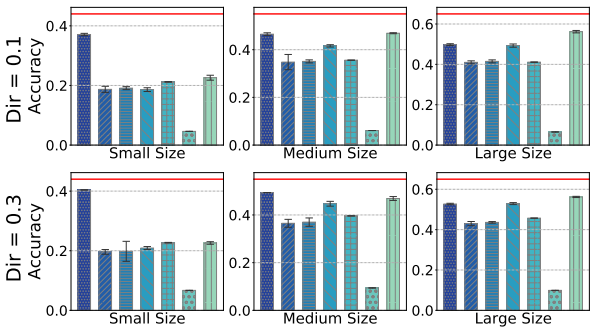
For the system performance, we evaluate the average waiting time, which is the delay between the time of the fastest and slowest client to complete local training, and the mean local update time of all clients in each federated round, an indicator of the compute overhead. As shown in Table 3, the average training time and mean local update time show consistent trends with the result among two testbeds, i.e., ArtFL outperforms other all baselines. This result shows that ArtFL can effectively alleviate the straggler issue. Moreover, we observe that the mean local update time of our edge testbed is significantly longer than the one obtained from the cloud server because the edge devices (e.g., Jetson Nano) are much weaker than the server GPUs. Yet, our approach still provides substantial improvement compared to baselines. Note that the training time estimation is accurate with a 0.33% and 0.39% error rate on average in our testbeds.

7 PUBLIC DATASETS EVALUATION

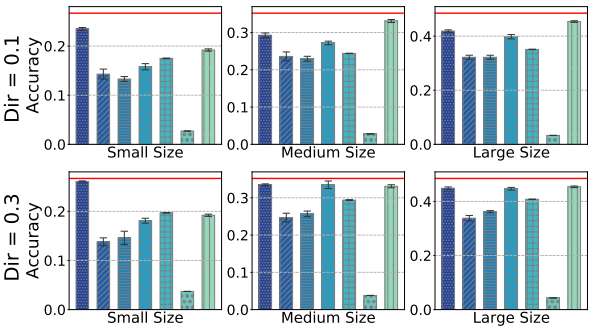
In this section, we evaluate ArtFL in a similar way with Sec. 6.3 on three public datasets (CIFAR10 [26], CIFAR100 [26], and Tiny-ImageNet [27]). Note that experiments in this section are conducted on the cloud server to validate the system's scalability. Specifically,



(a) CIFAR10 (10 clients)



(b) CIFAR100 (100 clients)



(c) Tiny (50 clients)

Figure 9: Accuracy comparison on three public datasets.

the clients of the three datasets are 10, 100, and 50, respectively. We use the same baselines mentioned in Sec. 5.1.

7.1 Accuracy and System Performance

First, the accuracy under two non-iid distributions (Dir = 0.1, and 0.3) are presented in Figure 9(a). Our approach outperforms other FL approaches under both Dir values in CIFAR10. For instance, when Dir = 0.1, our approach improved the absolute accuracy of FedAvg from 57.30% to 61.67%, 50.1% to 60.40%, and 31.8% to 48.3% in the original, medium, and small sizes (32×32, 24×24, and 16×16, respectively). In addition, we evaluate the performance of ArtFL on CIFAR100 (100 clients) and Tiny-ImageNet (50 clients),

Table 4: Average Waiting Time (Seconds) and Mean Local Update Time (Seconds) of public datasets when $Dir = 0.1$ and 0.3 distribution on the cloud server.

Dataset	Dir.	FedAvg	FedProx	FedDyn	BalanceFL	Ours
Average Waiting Time						
CIFAR10	0.1	56.59	62.22	54.69	81.72	**32.84**
	0.3	34.83	32.93	39.88	70.23	**20.72**
CIFAR100	0.1	15.56	16.55	20.80	41.08	**14.29**
	0.3	19.87	23.37	18.57	107.50	**19.17**
Tiny	0.1	52.30	54.23	57.71	237.72	**36.12**
	0.3	39.69	37.60	37.66	143.37	**33.51**
Mean Local Update Time						
CIFAR10	0.1	84.91	92.12	82.52	129.93	**55.84**
	0.3	101.75	97.53	95.28	172.56	**60.64**
CIFAR100	0.1	171.64	174.00	146.44	186.62	**134.41**
	0.3	192.61	182.43	165.66	261.33	**131.72**
Tiny	0.1	138.95	147.43	150.93	459.92	**103.82**
	0.3	138.33	147.64	156.66	569.13	**101.72**

Table 5: Ablation studies.

MST	D^3	*OFRS*	*RRS*	Accuracy	
				$Dir = 0.1$	$Dir = 0.3$
✓	✓	✓	×	63.19	80.35
✓	✓	×	✓	62.74	75.31
✓	✓	×	×	60.34	72.11
✓	×	×	×	54.30	66.31
×	×	×	×	46.30	57.10

which involve a larger number of clients and classes, respectively. Figure 9(b) and Figure 9(c) show our results exhibit a consistent trend with the experiments conducted on CIFAR10. For system performance, we observe that ArtFL performs the best, as shown in Table 4, compared to various baselines in both average waiting time and mean local update time. Specifically, on average waiting time, ArtFL only takes 32.84 seconds in the CIFAR10 dataset, yet the FedAvg and BalanceFL take 56.59 and 81.72 seconds, respectively. The results in mean local update time show the same trend, where our approach decreases the mean local update time by up to 40% compared to FedAvg. The above results show that ArtFL works well in different data non-iid distribution, demonstrating the robustness of distribution stragglers.

7.2 Ablation Studies

We present the ablation studies of ArtFL in Table 5, showing the result when multi-scale training (*MST*) and latency-based dynamic data dropping (D^3) are disabled on the CIFAR10 dataset with $Dir = 0.1$ and $Dir = 0.3$. Since the results on other datasets are similar, we present an ablation study conducted on the CIFAR10 dataset to evaluate the individual contributions of three key components, *Multi-Scale Training (MST)*, *Dynamic Data Dropping (D^3)*, and *Optimal Frame Resolution Selection (OFRS)*. Note that the results in comparing the *MST* and D^3 are the average among the three resolutions. Next, we follow the same constraint setting of the smart vehicle and compare compared *Optimal Frame Resolution Selection* with the *Random Resolution Selection (RRS)*, which randomly assigns resolutions to frames. The accuracy was evaluated under two different *Dir* conditions (0.1 and 0.3). When all components were equipped, the system achieved peak accuracies of 62.11 and 72.33 for *Dir* values of 0.1 and 0.3, respectively. In particular, the removal of *Dynamic Data Dropping* led to a more significant performance decrement, suggesting its substantial role in the system performance. Note that we randomly drop the data in the training process due to the information redundancy and keep the original data distribution. Meanwhile, the elimination of *Frame Resolution Selection* resulted in a nominal reduction in accuracy, indicating its positive but non-critical contribution.

Table 6: Comparsion in different Q and α.

Accuracy	Q_1	Q_2	$\alpha = 0$	$\alpha = 0.05$	$\alpha = 0.2$
Small	0.59	0.62	0.63	0.64	0.62
Medium	0.65	0.71	0.72	0.73	0.74
Large	0.69	0.73	0.73	0.76	0.75

8 RELATED WORK

Multi-Scale Training. Three different approaches have been proposed for multi-scale representation learning. The first approach is changing the input resolution by stochastic schemes [16, 56]. For example, MixSize [16] uses an adaptive stochastic training scheme to change the image size and the batch size in each optimization step. TRD [56] utilizes shape consistency for higher computational efficiency. However, these works can not be directly adopted to edge applications due to huge batch sizes, high memory usage, and a focus on centralized training. The second and third directions are geometric-based and learning-based methods [40]. In geometric-based methods, several solutions seek intrinsic geometric features such as edge and contour information [10]. As for learning-based methods, existing works focused on designing strategies of multi-scale structures that determine features' quality [18, 24, 33, 57]. For example, a multi-scale residual block [33] is proposed to force neural networks to learn different scale information via different kernel sizes. RANet [57] trains a hybrid network using different scales and branches. However, such approaches incur considerable additional compute overhead as the extra branches and varying kernel sizes.

Efficient On-Device Inference. To achieve efficient on-device inference, previous works are mainly focused on compressing the well-trained model through quantization [14, 20, 30], pruning [7, 22, 23, 31], and heterogeneous models [5, 13, 15]. For example, MobiSR [30] combines model compression techniques and a scheduler to achieve a desirable trade-off between image quality and processing speed. PruneFL [22] uses adaptive and distributed parameter pruning to optimize the model size for efficient inference. Moreover, FLAME [5] personalizes the model on heterogeneous devices to achieve efficient inference. SplitGP [15] split the full ML model into client-side and server-side components to accelerate device inference. TLFL [13] enables the on-board training of ML algorithms on IoT devices via transfer learning and TinyML techquine. Note that ArtFL can be easily combined with such post-processing techniques. Another approach is to achieve efficient inference using multi-models and an early exit scheme. For example, Chameleon [21] dynamically changes the networks to reduce the computation during inference based on the device computation resources. Branchynet [52] uses early exit branches to relieve the limitation in Chameleon, i.e., the constraint of multiple

independent model variants. Based on Branchynet, FlexDNN [12] considers the intrinsic dynamics of videos and dynamically adapts its model complexity to the difficulty levels of input frames. However, the aforementioned works only focus on centralized training and can not be directly deployed to the FL scenario due to a lack of consideration of the heterogeneity issues.

Heterogeneity Issues in FL. Heterogeneity issues can impact model convergence and diminish training efficiency in FL. To address this, the previous approach focuses on two key aspects, e.g., system heterogeneity and data heterogeneity. System Heterogeneity is primarily caused by the participating clients in heterogeneous computation and communication resources, leading to a catastrophic dropping in training efficiency. To improve the training efficiency, several approaches have been proposed, including knowledge distillation (KD) [6, 19, 37], partial training (PT) [6, 17, 18], client selection (CS) [29, 32, 46] and sample selection (SS) [2, 49]. In particular, KD-based methods aim to distillate a small (student) model from the big (teacher) model, while PT-based methods aim to reduce the computation by training subnets in each round. In addition, CS-based and SS-based methods mainly focus on dropping straggling clients and data points, respectively, to expedite the training efficiency. As for data heterogeneity is also known as non-i.i.d. data distribution caused by the data isolation in FL. There are mainly two types of FL [3], e.g., generic FL and personalized FL. Previous generic FL works such as FedAvg [41], FedProx [35], Scaffold [25], FedDyn [1] aim to generate a signal model. Personalized FL [3, 46, 58], targets to learn a personalized model in each client from meta-learning [36, 59] and multi-tasks learning techniques [39, 51, 58]. ArtFL is a generic FL approach and hence can easily work with the KD-, PT- and CS-, and SS-based methods, such as Oort [29] and Fedbalancer [49]. ArtFL optimizes the training efficiency in the sample level, which is orthogonal to the aforementioned methods.

In summary, to the best of our knowledge, ArtFL is the first study that adapts the multi-scale training to enable efficient inference on edge devices while addressing the heterogeneity issues in FL which can also improve the federated training efficiency.

9 DISCUSSION

More Complex Vision Tasks. The idea of ArtFL has the potential to be applied in various complex tasks. For instance, object detection tasks can obtain harnessing information by spanning multiple scales, ArtFL could substantially enhance the performance and accuracy of existing object detection algorithms. This multi-scale synthesis is poised to offer a more nuanced understanding of the visual data, thereby enriching the algorithm's ability to identify and classify objects with greater precision. However, the implementation of ArtFL is not without challenges. ArtFL requires to re-scale input to meet dynamic demands, which is not friendly for the low-resolution input or input with smaller objects. Such inputs struggle to adapt and maintain their integrity after the re-scaling process, potentially leading to a loss in detail or a diminished ability to recognize smaller objects. To address this limitation, a promising direction is selectively applying re-scaling operations and we can conserve the fidelity of the smaller or low-resolution elements while still benefiting from the advantages of multi-scale processing.

Joining/leaving of Clients. ArtFL accommodates the dynamic participation of clients, allowing for the seamless integration of new clients and the unobtrusive departure of existing ones. This level of flexibility, however, ushers in a set of sophisticated challenges, particularly evident in the phenomenon of knowledge loss that occurs when clients withdraw from the system. When clients leave, their unique contribution to the collective learning experience is at risk of being lost. This loss of information, or knowledge missing, as clients exit, poses a significant hurdle. It could potentially undermine the robustness of the system and the quality of the insights gleaned from it. The implications of such knowledge attrition may present a valuable direction for future research. Such research would be invaluable in fortifying the resilience of dynamic, client-based systems against the inevitable shifts in participant engagement, ensuring that the collective intelligence grows and is preserved over time.

Future Directions. Firstly, ArtFL can be extended to asynchronous FL systems. Although asynchronous FL systems can potentially reduce waiting time during training, they face challenges in model convergence because the slower clients may not upload their knowledge in time. ArtFL can address this issue by speeding up the slower clients' training duration. Next, ArtFL can be scaled to a larger number of clients in different applications. Due to the many co-running tasks in various applications, additional considerations must be considered in the optimal frame selection such as considering the concurrent tasks to design a more fine-grained frame selection framework. Lastly, as ArtFL primarily deals with data types rich in structured visual information like images and videos, we hope to extend ArtFL to other modalities with sequential data structures, such as speech or IMU data.

10 CONCLUSION

This paper proposes ArtFL, a novel FL framework designed to support dynamic inference requirements via data-utility-based multi-scale training, optimal frame resolution, and dynamic data dropping. Extensive experiments show that ArtFL delivers significant improvement over state-of-the-art baselines across public datasets and two real-world testbed. In the future, we will extend ArtFL to other data modalities and more complex tasks.

ACKNOWLEDGEMENT

This paper is supported in part by the Research Grants Council (RGC) of Hong Kong under Collaborative Research Fund (CRF) grants C4072-21G and the National Natural Science Foundation of China under grant 62032021.

REFERENCES

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. 2021. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263* (2021).
[2] Lingshuang Cai, Di Lin, Jiale Zhang, and Shui Yu. 2020. Dynamic sample selection for federated learning with heterogeneous data in fog computing. In *ICC*. IEEE.
[3] Hong-You Chen and Wei-Lun Chao. 2021. On Bridging Generic and Personalized Federated Learning. *arXiv preprint arXiv:2107.00778* (2021).
[4] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. 2020. Asynchronous online federated learning for edge devices with non-iid data. In *Big Data*. IEEE.
[5] Hyunsung Cho, Akhil Mathur, and Fahim Kawsar. 2022. Flame: Federated learning across multi-device environments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 3 (2022), 1–29.

[6] Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. 2022. Heterogeneous ensemble knowledge transfer for training large models in federated learning. *arXiv preprint arXiv:2204.12703* (2022).

[7] Yongheng Deng, Weining Chen, Ju Ren, Feng Lyu, Yang Liu, Yunxin Liu, and Yaoxue Zhang. 2022. TailorFL: Dual-Personalized Federated Learning under System and Data Heterogeneity. In *Sensys*. 592–606.

[8] Enmao Diao, Jie Ding, and Vahid Tarokh. 2020. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264* (2020).

[9] Jameson Dow. 2019. *Spurned by Tesla, NVIDIA's new Orin self-driving processor ups the game by 7x.* https://electrek.co/2019/12/18/nvidias-orin-self-driving-processor-7x-performance-xavier-tesla/

[10] Yiping Duan, Fang Liu, Licheng Jiao, Peng Zhao, and Lu Zhang. 2017. SAR image segmentation based on convolutional-wavelet neural network and Markov random field. *Pattern Recognition* 64 (2017), 255–267.

[11] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. 2021. Multiscale vision transformers. In *CVPR*. 6824–6835.

[12] Biyi Fang, Xiao Zeng, Faen Zhang, Hui Xu, and Mi Zhang. 2020. Flexdnn: Input-adaptive on-device deep learning for efficient mobile vision. In *SEC*. IEEE, 84–95.

[13] M Ficco, A Guerriero, E Milite, F Palmieri, R Pietrantuono, and S Russo. 2024. Federated learning for IoT devices: Enhancing TinyML with on-board training. *Information Fusion* 104 (2024), 102189.

[14] Kartik Gupta, Marios Fournarakis, Matthias Reisser, Christos Louizos, and Markus Nagel. 2022. Quantization Robust Federated Learning for Efficient Inference on Heterogeneous Devices. *arXiv preprint arXiv:2206.10844* (2022).

[15] Dong-Jun Han, Do-Yeon Kim, Minseok Choi, Christopher G Brinton, and Jaekyun Moon. 2023. Splitgp: Achieving both generalization and personalization in federated learning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[16] Elad Hoffer, Berry Weinstein, Itay Hubara, Tal Ben-Nun, Torsten Hoefler, and Daniel Soudry. 2019. Mix & match: training convnets with mixed image sizes for improved accuracy, speed and scale resiliency. *arXiv preprint arXiv:1908.08986* (2019).

[17] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. 2021. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *NeurIPS* 34 (2021), 12876–12889.

[18] Fatih Ilhan, Gong Su, and Ling Liu. 2023. ScaleFL: Resource-Adaptive Federated Learning With Heterogeneous Clients. In *CVPR*. 24532–24541.

[19] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. 2021. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE TMC* 22, 1 (2021), 191–205.

[20] Divyansh Jhunjhunwala, Advait Gadhikar, Gauri Joshi, and Yonina C Eldar. 2021. Adaptive quantization of model updates for communication-efficient federated learning. In *ICASSP*. IEEE, 3110–3114.

[21] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *SIGDC*.

[22] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. 2022. Model pruning enables efficient federated learning on edge devices. *IEEE TNNLS* (2022).

[23] Zhida Jiang, Yang Xu, Hongli Xu, Zhiyuan Wang, Chunming Qiao, and Yangming Zhao. 2022. Fedmp: Federated learning through adaptive model pruning in heterogeneous edge computing. In *ICDE*. IEEE, 767–779.

[24] Licheng Jiao, Jie Gao, Xu Liu, Fang Liu, Shuyuan Yang, and Biao Hou. 2021. Multi-Scale Representation Learning for Image Classification: A Survey. *IEEE TAI* (2021).

[25] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *ICML*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 5132–5143.

[26] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017).

[28] Gaurav Kumar and Durga Toshniwal. 2022. Neuron Specific Pruning for Communication Efficient Federated Learning. In *CIKM*. 4148–4152.

[29] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient Federated Learning via Guided Participant Selection. In *OSDI*. USENIX Association, 19–35.

[30] Royson Lee, Stylianos I Venieris, Lukasz Dudziak, Sourav Bhattacharya, and Nicholas D Lane. 2019. Mobisr: Efficient on-device super-resolution through heterogeneous mobile processors. In *MobiCom*. 1–16.

[31] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. 2021. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In *MobiCom*. 420–437.

[32] Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. 2022. PyramidFL: A fine-grained client selection framework for efficient federated learning. In *MobiCom*.

[33] Juncheng Li, Faming Fang, Kangfu Mei, and Guixu Zhang. 2018. Multi-scale residual network for image super-resolution. In *ECCV*. 517–532.

[34] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated learning on non-iid data silos: An experimental study. In *ICDE*. IEEE, 965–978.

[35] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *MLSys*, Vol. 2. 429–450.

[36] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. 2021. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623* (2021).

[37] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *NeurIPS* 33 (2020).

[38] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *ICCV*. 2980–2988.

[39] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619* (2020).

[40] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2022. Split computing and early exiting for deep learning applications: Survey and research challenges. *Comput. Surveys* 55, 5 (2022), 1–30.

[41] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. PMLR, 1273–1282.

[42] Aritra Mitra, Hamed Hassani, and George J Pappas. 2021. Online federated learning. In *CDC*. IEEE.

[43] John A Nelder and Roger Mead. 1965. A simplex method for function minimization. *The computer journal* 7, 4 (1965), 308–313.

[44] Anh Nguyen, Tuong Do, Minh Tran, Binh X Nguyen, Chien Duong, Tu Phan, Erman Tjiputra, and Quang D Tran. 2022. Deep federated learning for autonomous driving. In *IV*. IEEE, 1824–1830.

[45] NVIDIA. 2022. *Jetson Benchmarks.* https://developer.nvidia.com/embedded/jetson-benchmarks

[46] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. 2021. ClusterFL: A Similarity-Aware Federated Learning System for Human Activity Recognition. In *MobiSys*.

[47] DANNY SHAPIRO. 2022. *An Elevated Experience: XPENG Launches G9 EV, Taking Innovation Even Higher with NVIDIA DRIVE Orin.* https://blogs.nvidia.com/blog/2022/09/23/xpeng-g9-ev-innovation-drive-orin/

[48] Shuyao Shi, Jiahe Cui, Zhehao Jiang, Zhenyu Yan, Guoliang Xing, Jianwei Niu, and Zhenchao Ouyang. 2022. VIPS: real-time perception fusion for infrastructure-assisted autonomous driving. In *MobiCom*. 133–146.

[49] Jaemin Shin, Yuanchun Li, Yunxin Liu, and Sung-Ju Lee. 2022. FedBalancer: Data and Pace Control for Efficient Federated Learning on Heterogeneous Clients. (2022).

[50] Xian Shuai, Yulin Shen, Siyang Jiang, Zhihe Zhao, Zhenyu Yan, and Guoliang Xing. 2022. BalanceFL: Addressing Class Imbalance in Long-Tail Federated Learning. In *IPSN*. IEEE, 271–284.

[51] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated multi-task learning. *NeurIPS* 30 (2017).

[52] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*. IEEE, 2464–2469.

[53] Nicola Tonellotto, Alberto Gotta, Franco Maria Nardini, Daniele Gadler, and Fabrizio Silvestri. 2021. Neural network quantization in federated learning at the edge. *Information Sciences* 575 (2021), 417–436.

[54] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herve Jegou. 2019. Fixing the train-test resolution discrepancy. In *NeurIPS*, Vol. 32.

[55] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *TIP* (2004).

[56] Tianshu Xie, Xuan Cheng, Minghui Liu, Jiali Deng, Xiaomin Wang, and Ming Liu. 2021. Temporally Resolution Decrement: Utilizing the Shape Consistency for Higher Computational Efficiency. *arXiv preprint arXiv:2112.00954* (2021).

[57] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. 2020. Resolution adaptive networks for efficient inference. In *CVPR*. 2369–2378.

[58] Tianlong Yu, Tian Li, Yuqiong Sun, Susanta Nanda, Virginia Smith, Vyas Sekar, and Srinivasan Seshan. 2020. Learning context-aware policies from multiple smart homes via federated multi-task learning. In *IoTDI*. IEEE, 104–115.

[59] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. 2021. Personalized Federated Learning with First Order Model Optimization. In *ICLR*.

[60] Miao Zhang, Fangxin Wang, and Jiangchuan Liu. 2022. CASVA: Configuration-Adaptive Streaming for Live Video Analytics. In *INFOCOM*. IEEE, 2168–2177.

[61] Tianyue Zheng, Ang Li, Zhe Chen, Hongbo Wang, and Jun Luo. 2023. AutoFed: Heterogeneity-Aware Federated Multimodal Learning for Robust Autonomous Driving. In *MobiCom*. ACM.