IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 November 2024, accepted 13 December 2024, date of publication 17 December 2024,
date of current version 27 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3519254

RESEARCH ARTICLE

Benchmarking Federated Few-Shot Learning for Video-Based Action Recognition

NGUYEN ANH TU[1], NARTAY AIKYN[1], NURSULTAN MAKHANOV[1], ASSANALI ABU[1], KOK-SENG WONG[2], (Member, IEEE), AND MIN-HO LEE[1]

[1]Department of Computer Science, School of Engineering and Digital Sciences, Nazarbayev University, 010000 Astana, Kazakhstan
[2]College of Engineering and Computer Science, VinUniversity, Hanoi, Vietnam

Corresponding author: Nguyen Anh Tu (tu.nguyen@nu.edu.kz)

This work was supported by Nazarbayev University under the Faculty Development Competitive Research Grant Program (FDCRGP) with Grant No. 11022021FD2925.

ABSTRACT Few-shot action recognition aims to train a model to classify actions in videos using only a few examples, known as ''shots,'' per action class. This learning approach is particularly useful but challenging due to the limited availability of labeled video data in practice. Although significant progress has been made in developing few-shot learners, existing methods still face several limitations. Firstly, current methods have not sufficiently explored the effectiveness of 3D feature extractors (e.g., 3D CNNs or Video Transformers), thereby failing to exploit spatiotemporal dynamics in videos. Secondly, the need for a large video dataset to train the model in a centralized manner raises privacy concerns and results in high storage costs and communication overheads. Thirdly, the existing solutions based on local deployment lack the capability to benefit global prior knowledge from a wide variety of real-world action samples. To address these limitations, we propose a federated learning (FL) framework named FedFSLAR++ to collaboratively train few-shot learners with 3D feature extractors. Specifically, we perform few-shot action recognition tasks under FL settings, enhancing privacy protection while maintaining efficient communication and storage. Moreover, FL allows us to effectively learn meta-knowledge from a large set of action videos among heterogeneous clients. Within our framework, we establish a unified benchmark to systematically and fairly compare different components, including feature extraction, meta-learning, and FL for model update and aggregation. This type of benchmark is still lacking in the literature. Notably, we thoroughly examine six 3D CNN and Transformer models for extracting spatiotemporal video features needed to adapt to new tasks quickly during the meta-learning process. We further propose a hybrid feature extractor that combines the advantages of 3D CNNs and Transformers to produce strong video representations. Additionally, we explore three meta-learning paradigms and three FL algorithms to investigate their effectiveness and suggest the optimal choices for performance improvement. Results from extensive experiments on four action datasets verify the robustness of the FedFSLAR++ framework. Our comprehensive study provides a solid foundation for future research advancements in action recognition.

INDEX TERMS Human action recognition, few-shot learning, federated learning, representation learning, few-shot action recognition.

I. INTRODUCTION

In today's era, the rapid proliferation of camera devices, coupled with the widespread adoption of social media platforms, has led to a remarkable explosion in video data volume. This surge has accelerated the growth of human action recognition (HAR) as an active research area in computer vision. The capability to recognize complex human actions has a profound impact on societal applications, spanning from advanced surveillance systems to dynamic web-video search

The associate editor coordinating the review of this manuscript and approving it for publication was Tai Fei.

and retrieval, from patient monitoring to insightful sports analysis, and from enhanced human-computer interaction to innovative technological advancements. Recent years have marked significant advancements in video-based HAR owing to the advent of Deep Learning (DL) architectures and the availability of action datasets [1]. Nonetheless, obtaining robust feature learning in action models necessitates substantial quantities of video data collected from diverse sources and considerable computational resources. Existing DL methods [2], [3] largely rely on centralized training, which entails high storage expenses and communication overheads to transmit local data from clients to a central server. Additionally, human bodies often expose various personal attributes, including identity, gender, age, and movement patterns. This sensitive information can be easily exploited for unauthorized analyses, leading to privacy breaches. Consequently, Federated Learning (FL) [4] has emerged as an effective decentralized strategy, enabling collaborative learning of a shared model. FL aggregates local models computed on individual devices to train a globally shared model on a central server, enhancing privacy protection and communication efficiency by eliminating the need to transmit sensitive video data. Also, FL enables us to train models using more diverse datasets sourced from heterogeneous devices, thereby allowing the acquisition of more intricate patterns and greater generalizability.

Both centralized learning (CL) and FL demand sufficiently labeled data, typically requiring hundreds of samples per class to achieve optimal training performance of deep neural networks [5]. Yet, manually annotating a large volume of videos is costly and laborious. Besides, curating new video datasets for end-users or organizations such as airports, schools, banks, and hospitals is difficult whenever encountering novel action classes. In many real-world scenarios, novel action classes emerge over time, posing complications in acquiring adequately annotated data to tackle these evolving challenges. For example, the airport needs to periodically update suspicious actions, which have never been seen before, from the passengers for security purposes. However, effective preparation of a new training dataset proves challenging, given the diverse nature of human actions. In practice, video data collected from various devices and organizations often lacks completeness, severely limiting the practicality and scalability of machine learning models. Moreover, disparities in the quantity of data available among different organizations can lead to biases within learned models and reduce their ability to generalize effectively. Thus, Few-Shot Learning (FSL) [6] has emerged as a promising and practical direction for training machine learning models to recognize novel action classes using only a small number of support samples.

FSL assesses a model's generalization capacity to adjust to new tasks, presenting a formidable challenge due to the scarcity of available data for model adaptation [7]. Lately, significant progress has been achieved in addressing this issue through the concepts of meta-learning, which imitates human learning capabilities by leveraging prior knowledge and experiences from past tasks. Meta-knowledge is acquired by learning action videos from base classes and generalizing them to novel classes with a few labeled videos. Here, base and novel classes must be clearly different. Recent works on FSL [8] have demonstrated that features obtained through effective embedding networks play a crucial role in fast adaptation. Existing methods for few-shot action recognition [9], [10] primarily rely on 2D-CNN backbones to capture frame-level features representing action appearance. These features are then utilized in conjunction with metric learning and temporal alignment techniques to calculate the similarity between different videos for classification. However, employing frame-level features does not adequately capture the temporal correlations present among video frames. This limitation is particularly pronounced in few-shot scenarios, where relying solely on appearance cues from limited examples might hinder the model's performance. Some studies [11] have employed more robust spatiotemporal backbones (e.g., 3D-CNNs [2], [12]) for feature embedding, aiming to integrate temporal dynamics into video representations. Despite these efforts, the performance of such methods has not met expectations. 3D feature extractors remain relatively underexplored in the context of few-shot action recognition.

Moreover, existing methods usually require a large amount of centralized data to train the FSL model on a single machine. Consequently, they pose privacy threats, incur high data storage costs, and face scalability issues, as discussed above. To protect privacy and address these challenges, local deployment at each institution is a feasible solution. However, with centralized deployment, each institution can easily struggle to train an FSL model capable of accurately classifying unseen actions. This limitation stems from the inability to leverage prior knowledge extracted from diverse sources of video samples. Therefore, it is crucial to instantiate FSL within the FL framework, enabling it to handle new tasks with only a few samples. This issue is termed Federated Few-Shot Learning (FedFSL) [13], leveraging FL to preserve privacy, decrease communication costs, and enhance the overall practicality. Nonetheless, conducting FSL in FL environments is challenging due to variations in task domains among different client types. For example, actions, e.g., shoplifting, fighting, and stealing, in public areas typically differ from daily indoor actions, e.g., cleaning, sleeping, and cooking. Furthermore, the considerable intra-class variability, as opposed to limited inter-class variability [14], poses a significant obstacle to achieving optimal model accuracy in FL.

Currently, there is limited research on FedFSL. Although it has shown satisfactory performance across several tasks in the literature, such as image classification [15], [16], language processing [17], and sound classification [18], its effectiveness in addressing video-based HAR remains

uncertain. To the best of our knowledge, our preliminary work [19] represents the first study to explore FedFSL for video-based HAR. This paper is an extension of [19], which is built upon the aforementioned insights. Specifically, we address the few-shot action recognition problem by introducing an innovative FL framework, namely FedFSLAR++, as illustrated in Fig. 1, which meta-trains the global embedding network across clients. Video representations provided by the embedding network are then inputted into the classifiers on each client to predict human actions. This is motivated by the understanding that developing an effective video representation necessitates leveraging the collective abundance and heterogeneity of video data available from various institutions (e.g., banks and airports). During training, each client stores local video data with distinct tasks and data distributions. We investigate various 3D embedding networks to extract video features and undertake meta-training to preserve generalization abilities to new tasks within these distributions. Notably, we employ different federated adaptations of FSL algorithms. In the iterative FL process, each device updates the model using its local dataset and transmits the updated model to the server for aggregation. Our FL framework is designed to answer the following questions: How can we meta-train the shared embedding network (or feature extractor) under FL settings to maintain robustness across heterogeneous data distributions among clients? What 3D feature extractors are suitable for FedFSL? How do different FSL and FL algorithms impact the recognition performance of our framework? Exploring these questions is both intriguing and deserving of a comprehensive investigation. In essence, this paper seeks to conduct a thorough analysis of each component within the FedFSL framework and elucidate good practices for developing a state-of-the-art few-shot action recognition system.

Notably, under the FedFSLAR++ framework, we create a benchmark to facilitate deeper exploration in the field and enable fair comparisons. Our experiments employ datasets categorized into two distributions: identical and independent distribution (IID) and non-identical and independent distribution (non-IID), aiming to mimic realistic FL systems. Through extensive experimentation on benchmark datasets, we demonstrate the efficacy of discriminative video representations. These representations can be transferred and fine-tuned for new tasks involving novel classes, thereby bridging the gap between CL and FL. However, in some cases, distributedly training feature backbones from scratch with random weights might result in decreased accuracy. Meanwhile, using feature backbones pre-trained on external data with suitable settings, FL can achieve comparable or superior accuracy compared to CL. These findings underscore the importance of exploring robust representation learning through pre-training for few-shot action recognition. Also, our findings align with recent studies in image domain [20], [21], highlighting pre-training as a viable method for acquiring robust feature representations, particularly beneficial in FSL scenarios. Furthermore, learning meta-knowledge from video data with limited variation poses challenges, particularly in the FL setting, where clients have only a few classes. These challenges highlight avenues for further research in FL and FSL, particularly within the domain of action recognition tasks. Our main contributions are summarized as follows:

- We propose a unified FL framework for few-shot action recognition. The development of this benchmarking framework would greatly enhance future research endeavors, facilitating systematic comparisons on multiple dimensions for action recognition, such as FSL, FL, and feature extraction.
- We conduct a comprehensive empirical study on various types of 3D feature extractors (Fig. 1), including 3D-CNNs (I3D [2], R3D [22], R(2+1)D [22], Slow [23]), and Video Transformer (MViT [24]). Additionally, we propose a hybrid model combining the advantages of Transformers and 3D-CNNs, named STCA.
- We delve into different FSL algorithms, including ProtoNet [25], RelationNet [26], and Reptile [27], and present thorough experiments and analysis. Notably, we introduce an attention-based RelationNet customized for our STCA embedding network. Our empirical findings suggests that metric-based algorithms, such as ProtoNet and RelationNet, are better suited for meta-training 3D feature backbones compared to optimization-based approaches like Reptile.
- We especially investigate the impact of non-pre-training and pre-training on the FedFSL performance. We show that pre-trained MViT and SCTA outperform CNNs by leveraging self-attention to exploit meta-knowledge from pre-training data. Without pre-training, MViT underperforms compared to CNNs and SCTA due to the absence of key inductive biases, while SCTA maintains competitive performance, showcasing its adaptability.
- We systematically compare three popular FL algorithms, including FedAvg [28], FedProx [4], and Moon [29] under various realistic settings for FedFSL.
- Our benchmark reveals that combining pre-trained feature extractors with suitable FL settings can achieve state-of-the-art performance on challenging datasets of few-shot action recognition. This success stems from the transferability of meta-knowledge from both external data and FL clients' local data during aggregation, with pre-trained models facilitating knowledge exchange and improving the global model's adaptability to new tasks.

This paper extends our preliminary work [19] in the following aspects: (1) We explore more feature extractors ([19] only discussed 3D CNNs). (2) We investigate more meta-learning processes ([19] only presented ProtoNet). (3) We extend our study to propose a new hybrid feature extractor, which can be meta-trained using our improved version of RelationNet. (4) We explore additional FL methods for model updates and aggregation. (5) We provide a more comprehensive study and extensive comparison with current

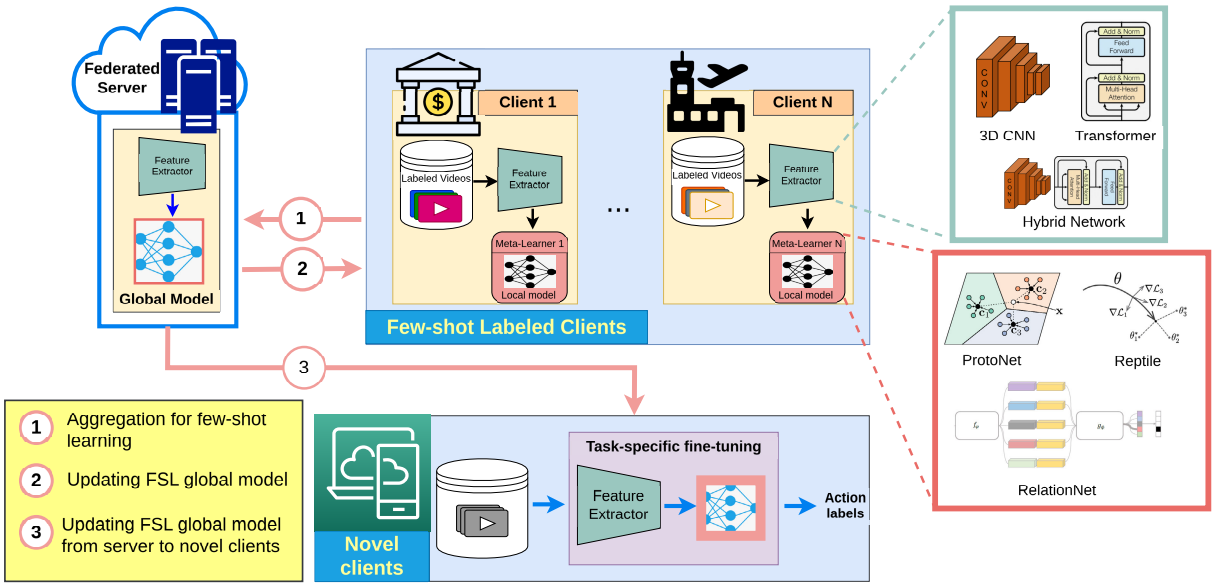

FIGURE 1. Overall system pipeline for federated few-shot action recognition.

methods. The structure of this paper is as follows: Section II reviews related works. Section III outlines our federated few-shot action recognition framework and details its main components, including feature extraction, FSL methods, and FL algorithms for model update and aggregation. Section IV presents and discusses experimental results on benchmark datasets. We thoroughly analyze each component of the FedFSL framework and highlight best practices for obtaining state-of-the-art performance. Lastly, Section V provides the conclusions.

II. RELATED WORKS

A. FEATURE EXTRACTION FOR ACTION RECOGNITION

Feature extraction is an important step in human action recognition because it helps capture the critical action characteristics, improving the recognition system's ability to differentiate between actions. Traditionally, video classification was dominated by hand-crafted features [30], [31], [32] due to their efficacy in handling background noises. For example, one of the most popular methods is Dense Trajectories [31], [32], which extracts human trajectories directly from a video without the need for body tracking and can subsequently be used in classification models like bag-of-visual words [33]. However, hand-crafted features are computationally expensive and challenging to deploy. Later, the rapid growth of deep learning and parallel computing technologies motivated researchers to develop deep feature extraction methods for vision tasks. Deep features for action recognition can be divided into two groups: frame-wise features [34] and spatiotemporal features [35]. The features of the former group are typically extracted by 2D-CNNs and temporally aggregated using different connectivity techniques, such as late fusion [36] and early fusion [37], to generate a spatiotemporal representation for video action recognition. More advanced aggregation methods, such as LSTM [38] and ConvLSTM [39], have since been developed. However, the video representation learned in this way does not capture motion information well. The latter feature extraction methods have been devised by using 3D-CNNs [2], [22], [23] to better exploit the temporal relationship among video frames. Subsequently, with the rise of attention mechanisms in deep learning, Transformer models [40], [41] have gained significant attention for extracting robust video features. On the other hand, inspired by the success of the Large Language Model (LLM), recent studies [42], [43] have increasingly focused on integrating language semantics into vision models to strengthen visual representations. In this paper, we explore different types of feature extractors and analyze their effect on the performance of few-shot action recognition.

B. FEW-SHOT ACTION RECOGNITION

Few-shot action recognition involves identifying new actions based on only a handful of examples. In this subsection, we explore the major approaches developed for few-shot action recognition. One of the prominent strategies is metric-based meta-learning, which focuses on creating a versatile metric space that can effectively compare videos depicting different actions. In many of these approaches [10], [44], [45], [46], [47], [48], [49], frame-level features are extracted using 2D-CNNs. These features are then used with temporal alignment methods to measure the similarity between query and support videos by calculating distances within the learned metric space. The compound memory network (CMN) introduced by [10] utilizes multiple constituent keys and a multi-saliency embedding algorithm to enhance action recognition. Cao et al. presented OTAM [44], an online temporal attentive module that dynamically selects and aligns

the most informative frames from the support set. In another notable work, TRX [45] employs CrossTransformers to generate class prototypes from relevant sub-sequences of support videos, facilitating comparisons between video tuples with different frame counts. The FSL performance of CMN, OTAM, and TRX is often restricted by their inflexible matching strategy, particularly when dealing with complex actions composed of multiple sub-actions in varying sequences. This inflexibility makes it challenging to accurately identify the correct correspondences in such scenarios [48]. To overcome this limitation and improve performance, several studies [46], [48] have introduced hybrid methods for temporal alignment that employ more flexible metrics. ITA-Net [46] incorporates a hybrid spatial and feature channel context to learn implicit temporal alignment for efficient estimation of video similarity. Wang et al. [48] proposed the HyRSM++, which integrates two types of relation scores derived from both global and local features to enhance action recognition. Another way to boost FSL performance is by incorporating additional data modalities, thereby providing a more powerful video representation. For example, CLIP-FSAR was proposed in [49] that utilizes the multimodal capabilities of the CLIP model and employs a video-text contrastive objective to adapt it for the few-shot action recognition task.

Many methods focus on 2D feature extraction, but they often struggle to effectively capture the temporal dependencies among video frames. This limitation can lead to misalignment, particularly when identifying complex actions involving subtle motion variations. Fewer approaches [50], [51], [52], [53], [54] utilize spatiotemporal models to obtain comprehensive video-level features. For example, TSL [50] leverages the R(2+1)D [22] architecture to extract features and uses text-based queries to find additional videos that can augment the training set. Techniques like TARN [51] and CMOT [52] use C3D [35] to generate feature vectors and handle variable-length video comparisons through deep distance metrics. Cao et al. [55] tackled the challenges of spatiotemporal feature matching and alignment in few-shot action recognition by introducing a specialized Transformer search method to optimize spatial and temporal attention, along with a non-parametric prototype alignment strategy to effectively handle motion variability. Liu et al. [53] developed a fusion mechanism that can be integrated with 3D feature extractors, significantly improving the key spatial and temporal regions in video representation. Notably, the authors demonstrated the superiority of 3D feature extractors over 2D ones under identical experimental conditions. To further enhance the performance of 3D backbones, SAFSAR [54] employs the multi-modal fusion model, which extracts discriminative spatiotemporal features by 3D Transformer while leveraging textual semantics.

Additionally, there are approaches aimed at expanding the available data through data augmentation or by generating new samples with generative models. For example, Proto-GAN [56] employs a conditional GAN to produce more samples for each class in the support set, enhancing the diversity and quantity of training data. AMeFu-Net [57], on the other hand, combines depth and visual information and uses temporal asynchronization to augment the data. Although these generative-based methods can address data scarcity issues, they are computationally intensive and fail to achieve comparable FSL performance.

In spite of their success, these methodologies are primarily centralized. The reliance on centralized processing limits their applicability in scenarios where data cannot be easily shared across different locations due to privacy concerns. Moreover, meta-learning has emerged as the most popular FSL solution, largely due to its remarkable ability to extract knowledge from diverse tasks and enable quick adaptation when encountering newly unseen tasks. However, few-shot action recognition methods mostly focus on metric-based meta-learning, with limited direct comparisons to other meta-learning types, such as optimization-based methods like Reptile [27]. To address this gap, this paper systematically explores different meta-learning methods under consistent experimental conditions and provides a comprehensive analysis.

C. FEDERATED ACTION RECOGNITION

Federated action recognition is at the forefront of privacy-focused strategies in computer vision, utilizing FL and distributed training to safeguard sensitive data. This paradigm shift moves away from centralized data collection, relying instead on edge devices or servers to uphold privacy while developing a global model. This decentralized approach is particularly vital in privacy-sensitive and bandwidth-limited environments, making it highly applicable in fields like surveillance and healthcare. Federated action recognition marks significant progress in creating secure and efficient action recognition models. For example, Doshi and Yilmaz [58] and Yuan et al. [59] investigated distracted driver activities using FedAvg algorithms [28] with 2D-CNN models on RGB video data. Doshi and Yilmaz [58] further implemented the FedGKT algorithm [60] to reduce resource usage on edge devices, while Yuan et al. [59] proposed a peer-to-peer FL to improve learning efficiency. With their advanced designs in FL settings, these methods achieved accuracy comparable to that of centralized models. Besides, FL methods have been developed in [61] and [62] for action recognition from depth cameras. Psaltis et al. [61] introduced a method for integrating depth and 3D flow into multi-modality fusion schemes and federated aggregation mechanisms for cross-domain knowledge transfer in a distributed manner. Meanwhile, Guo et al. [62] designed an FL framework named FSAR for 3D skeleton-based action recognition. Using the heterogeneous human topology graph structure and multi-grain knowledge distillation, FSAR outperforms FedAvg on several skeleton-based benchmarks. Nonetheless, action recognition from depth cameras often perform poorly in outdoor environments [63]. Beyond camera sensors, several action recognition methods [64], [65] have

investigated FL paradigms for other sensor types. Xiao et al. [64] explored FL for human action recognition (HAR) using wearable sensors, developing a hybrid CNN-LSTM-based attention algorithm combined with homomorphic encryption that outperformed existing HAR algorithms across various datasets. Subsequently, Ouyang et al. [65] advanced FL for HAR by proposing ClusterFL, a system that leverages clustering to enhance model accuracy and communication efficiency. ClusterFL demonstrated comparable performance to the centralized learning across a wider range of sensors, including ambient, wearable, and depth cameras.

While existing federated action recognition methods have demonstrated promising performance, they are primarily designed for specific tasks (e.g., driver activity recognition) or specialized data types (e.g., 3D skeleton and depth images). These approaches are often limited to certain environments, such as in-car or indoor settings. Moreover, their experiments are typically conducted on small to medium-sized datasets with a limited number of action classes. A significant gap remains in establishing benchmarks that enable fair and systematic comparisons of general action recognition models within federated learning (FL) settings. Additionally, current methods do not adequately address key challenges such as data scarcity or the training of models on newly unseen action classes, which is the main focus of this paper. We also provide a comprehensive benchmark for general action recognition, utilizing challenging datasets that encompass a wide range of action classes across various types and diverse contexts.

D. FEDERATED FEW-SHOT LEARNING

FSL has made significant strides with the FL integration, enabling innovative approaches to train models with limited labeled data across privacy-aware settings. This section highlights key studies contributing to this emerging field. Fan and Huang [13] were pioneers in proposing adversarial training and client model optimization to create a more discriminative feature space for better representation of unseen data samples. Wang et al. [16] introduced a novel FedFSL framework that uses two separately updated models and specific training strategies to mitigate the challenges posed by global data variance and local data insufficiency. From an application standpoint, the performance of few-shot image classification under the FL framework has been explored in [66], [67], and [68]. Chen et al. [69] developed the FedMeta-FFD framework to improve mechanical fault diagnosis in the industrial Internet of Things. Cai et al. [17] proposed FeS, a framework enabling practical few-shot NLP fine-tuning on federated mobile devices. Furthermore, Hoang et al. [18] created the F2LCough framework for cough sound classification to aid in diagnosing and treating respiratory diseases. While these studies have explored FedFSL for various simple tasks involving different data types such as images, text, and audio, its application to more complex vision tasks like video-based action recognition remains an open area for research.

III. METHODOLOGY

In this section, we first formulate the FSL problem under FL settings and describe the general FL algorithm for benchmarking few-shot action recognition tasks. Then, we present the background knowledge of each key component in our benchmark framework, including feature extraction, episodic meta-learning, and FL.

A. PROBLEM DEFINITION

Federated few-shot action recognition seeks to identify action classes with limited video examples from a group of K clients $\{\mathcal{C}_k\}_{k=1}^{K}$ and a federated server $\mathbb{S}$. In the client $\mathcal{C}_k$, we categorize actions into two distinct sets: base classes ($\mathcal{X}_b^{(k)}$), which are used for training with numerous labeled videos, and novel classes ($\mathcal{X}_n^{(k)}$), which the model has not seen before and has only a few videos. The meta-learning process is then conducted through episodic training, with episodes in client $\mathcal{C}_k$ being drawn from ($\mathcal{X}_b^{(k)}$). Each episode, corresponding to a specific task, consists of a support set $\mathcal{S}^{(k)} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{M\times P}$ and a query set $\mathcal{Q}^{(k)} = \{\mathbf{X}_j\}_{j=1}^{L}$. The support set includes labeled examples from M different classes, with P samples per class, while the query set includes L unlabeled samples used for classification within that episode. This problem is described as M-way P-shot.

In an episode for client $\mathcal{C}_k$, the aim is to train a model $\Theta^{(k)}$ to classify the query set $\mathcal{Q}^{(k)}$ using the support set $\mathcal{S}^{(k)}$. This involves minimizing the loss between the predicted labels for $\mathcal{Q}^{(k)}$ and the actual labels. In the FL framework, local models $\{\Theta^{(k)}\}$ are sent to a federated server $\mathbb{S}$, where they are aggregated into a global model $\{\Theta\}$, which is then distributed back to the clients for further training rounds. In this way, We train the global model $\mathbb{S}$ using data from all clients without directly exchanging data between them, hence preserving the data privacy. During testing, we apply the learned model Θ to fine-tune new tasks for the novel clients. In this scenario, the shared model $\{\Theta\}$, representing the meta-knowledge from base classes, can be effectively generalized to classify videos in novel action classes with a few labeled samples. The overall architecture of our proposed method is depicted in Fig. 1.

B. FEDERATED FEW-SHOT LEARNING ALGORITHM

Building on the above problem definition and our proposed benchmark architecture, we present a general algorithm for federated few-shot action recognition. This algorithm is essential for advancing the research presented in this paper and future studies in related fields. The procedure is detailed in Algorithm 1, which outlines the operations across multiple communication rounds to coordinate the synchronized optimization of individual client models. Each client starts by sampling episodes from its local dataset and then improves its local model through episodic training. After these local updates, the client models are aggregated to create a new global model, taking into account the proportion of local data each client possesses. This algorithm allows

Algorithm 1 FedSLAR++ Algorithm

Input : Number of communication rounds T, Number of clients K, Dataset $(\mathcal{X}_b, \mathcal{X}_n)$
Output : Global Model Θ
Initialize : Θ_0
for $t \leftarrow T$ **do**
  for *each client k in parallel* **do**
    $\Theta_t^{(k)} \leftarrow$ **ClientUpdate**(Θ_t)
  end
  Clients send model parameter $\{\Theta_t^{(k)}\}_{k=1}^{K}$ to server for aggregation: $\Theta_{t+1} = h_{FL}(\Theta_t^{(1)}, \Theta_t^{(2)}, \ldots, \Theta_t^{(K)})$, where $h_{FL}(\cdot)$ indicates the update formulas of different FL algorithms, including FedAvg, FedProx, and MOON.
end
Return Θ_t
ClientUpdate()
:
  Input: global model from previous round Θ_t
  Output: updated local model $\Theta_t^{(k)}$
  Sample a set of episodes with support and query samples from $\mathcal{X}_b^{(k)}$:
    $\{(\mathcal{S}_1, \mathcal{Q}_1), (\mathcal{S}_2, \mathcal{Q}_2).., (\mathcal{S}_m, \mathcal{Q}_m)\}$
  In each training episode $(\mathcal{S}_i, \mathcal{Q}_i)$, compute feature maps of query and support samples $f_{\Theta_t^{(k)}}(\mathcal{S}_i), f_{\Theta_t^{(k)}}(\mathcal{Q}_i)$
  Optimize $\Theta_t^{(k)}$ w.r.t the classification loss $\mathcal{L}(f_{\Theta_t^{(k)}}(\mathcal{S}_i), f_{\Theta_t^{(k)}}(\mathcal{Q}_i)$ using episodic training process like ProtoNet, RelationNet, or Reptile.
  Return $\Theta_t^{(k)}$

TABLE 1. Summary of notations used in this paper.

Notations	Description
$\mathcal{X}_b, \mathcal{X}_n$	Base class set and novel class set
$\mathcal{S}, \mathcal{Q}$	Support set and query set for meta-learning process
Θ	Global parameter of the feature extractor for all clients
Θ^k	Local parameter of the feature extractor at the client k
$\mathbf{X}$	Input video with T frames
$\mathbf{F}$	Feature map contains local feature embeddings
$\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$	Learnable projection matrices of the Transformer encoder to linearly convert feature map $\mathbf{F}$ into query, key, and value matrices, respectively
$f_\Theta(\cdot)$	Feature extraction function parameterized by Θ
$\hat{\mathbf{X}}$	Video representation
$\mathcal{D}$	Dataset includes video samples from all clients
$\mathcal{D}_k$	Local dataset contains video samples from a client k
η	Learning rate for the FedProx algorithm
λ	Proximal term coefficient for the FedProx algorithm
τ	Temperature parameter for the MOON algorithm

us to explore various feature extractors, different federated learning algorithms, and few-shot learning processes, all key components of our proposed benchmark architecture shown in Fig. 1. The benchmark suite of these components is summarized in Table 2. In the subsequent sections, we provide an overview of existing methods and detail our proposed solutions and improvements within the context of few-shot action recognition. Additionally, notations used in this paper are summarized in Table 1.

C. FEATURE EXTRACTION

Extracting video features plays a pivotal role in few-shot action recognition. A good feature is essential for quickly adapting to new tasks during the meta-learning process. Hence, an effective embedding network is required to learn feature representations from video sequences. Current methods [9], [10], [45], [48], [70] primarily employ 2D-CNN as backbones to capture frame-level features. These are subsequently processed by metric learning and temporal alignment techniques to estimate the similarity between different videos for classification. Although significant progress has been made, these few-shot action recognition methods neglect the sequential ordering of frames. Their inability to capture temporal information between adjacent frames may hinder the acquisition of strong feature representations. As such, this study centers on embedding networks like 3D-CNNs [22] and Transformers [41], renowned for their ability to effectively exploit spatiotemporal correlations between frames, leading to improved performance in video classification tasks. These feature extractors have not been widely used for few-shot action recognition and merit further exploration. Although certain prior studies [11], [52] have incorporated 3D-CNNs, such as C3D [35], their performance has been limited and inferior to that of 2D-CNN counterparts.

Spatiotemporal feature extractors generally fall into two categories: 3D-CNNs and Video Transformers. 3D-CNNs, comprising convolutional layers with 3D kernels [22], inflated 3D kernels [2], or (2+1)D kernels [22], are designed to capture spatiotemporal dynamics from video frames. Feature maps undergo dimensionality reduction via temporal pooling layers before being fed into fully connected layers for classification. Meanwhile, the Video Transformer initiates by embedding the raw video data into a sequence of compact representations, directly serving as tokens. These tokens are then fed into self-attention mechanisms and feedforward neural networks. The model processes the sequence of tokens to capture both spatial and temporal information, ultimately producing a classification output based on the learned representations. One of the most effective Video Transformer models is Multiscale Vision Transformers (MViT) [41]. This Transformer model captures hierarchical features across multiple scales, enabling a nuanced understanding of videos' spatial and temporal dimensions. MViT employs a series of transformer blocks that progressively decrease the spatial resolution while increasing the feature dimension. The MViT architecture, by focusing on relevant features at various scales and complexities, enhances its video classification accuracy.

While these spatiotemporal models have demonstrated success in HAR, their application to few-shot action recognition remains largely unexplored. Furthermore, there are notable limitations within these models that could hamper their efficacy in producing strong representations for FSL tasks. On the one hand, 3D-CNNs process input data using local receptive fields, potentially limiting their power to capture long-range dependencies within videos. Furthermore, their reliance on fixed-size convolutional filters may hinder

TABLE 2. Summary of methods for benchmarking.

Feature extractor (f_Θ)	R3D [22], R(2+1)D [22], I3D [2], Slow [23], VTN [40], MViT [24], STCA
Episodic training process	Reptile [27], ProtoNet [25], RelationNet [26], Attention-based RelationNet
FL algorithm	FedAvg [28], FedProx [4], Moon [29]

Note: Underline denotes the methods proposed in this paper.

adaptation to videos' varying spatial and temporal scales. On the other hand, Transformers lack certain beneficial inductive biases present in CNNs, necessitating a lot of data and computational resources to compensate for this deficiency. Despite their scalability, with both architectures capable of handling larger models and datasets, Transformers have shown superior adaptability and scalability owing to their self-attention mechanism, allowing them to accommodate diverse tasks and data distributions more effectively.

1) SPATIOTEMPORAL CONVOLUTION WITH ATTENTION NETWORK (STCA)

Inspired by the recent achievements of a hybrid model [71] in image classification, which addressed the constraints of both CNNs and Transformers, we introduce an embedding network for few-shot action recognition. This network combines the advantages of spatio-temporal convolutions and self-attention mechanisms, referred to as STCA, as depicted in Fig. 2. Our hybrid network is designed with the understanding that convolutional layers excel in generalization due to their robust inductive bias. Meanwhile, attention layers boast superior model capacity, which is particularly beneficial for large action datasets. The fusion of convolutional and attention layers offers improved generalization and capacity, which are essential for FSL tasks.

In Fig. 2, given the input video with T frames (denoted as $\mathbf{X} \in \mathbb{R}^{T \times D_{in}}$), we illustrate our process of extracting the feature. Our feature extractor consists of 3D convolutions and the Transformer encoder, where we leverage the 3D-CNN architecture, SlowFast [23], to generate feature embeddings for Transformer. These embeddings are instrumental in locally capturing temporal interactions across the neighboring frames. Notably, we employ the Slow pathway of SlowFast, which entails a temporally strided 3D ResNet. This Slow model comprises several components, including a convolutional layer ($conv_1$) with pooling ($pool_1$) and four residual blocks (res_2, res_3, res_4, res_5). Each residual block incorporates convolutional layers with skip connections. Following [23], We use 2D convolutional filters from $conv_1$ to res_3, while temporal convolutions (underlined in the Fig. 2) are used in res_4 and res_5. The Slow output is the feature map with the shape $\{T \times S^2 \times D\}$, where S is the spatial height and width, and D denotes the channel size. Then, we use spatial pooling and reshaping techniques to obtain T local feature embeddings denoted as $\mathbf{F}_i \in \mathbb{R}^D, i = \{1..T\}$. These local embeddings can be formed as a matrix $\mathbf{F} \in \mathbb{R}^{T \times D}$. Here, the Slow network can be expressed as $f_{Slow}(\mathbf{X})$ that converts the input video $\mathbf{X}$ into the feature map $\mathbf{F}$. Note that we do not use the Fast pathway of SlowFast to prevent an increase in model complexity and the need for additional input frame sequences with higher frame rates.

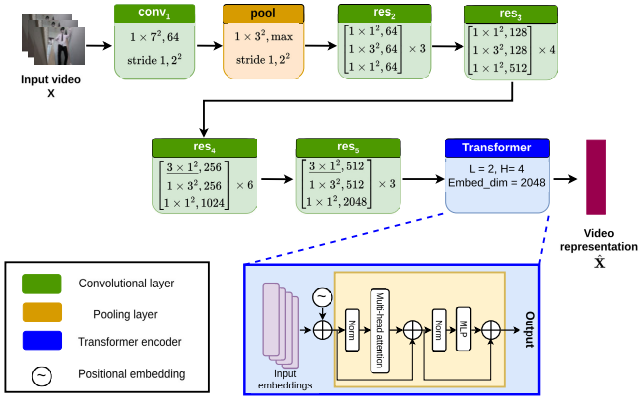

FIGURE 2. STCA feature extraction process.

Also, unlike the SlowFast architecture, we replace the temporal pooling at the end with self-attention operations from the Transformer encoder to aggregate T local feature embeddings $\mathbf{F}_i$ into the robust video representation. The embeddings are precisely mapped linearly onto the embedding spaces corresponding to queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$ as follows: $\mathbf{Q} = \mathbf{F}\mathbf{W}_q, \mathbf{K} = \mathbf{F}\mathbf{W}_k, \mathbf{V} = \mathbf{F}\mathbf{W}_v$, where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^D \times d$ are learnable projection matrices. Subsequently, the self-attention can be calculated using the following formula:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \quad (1)$$

The Transformer encoder consists of L multi-head attention (MHA) layers. Within each MHA layer, self-attention is executed across H independent heads, each of which maps the input embeddings into different representations. These representations are then concatenated and passed through the normalization and MLP sub-layers to generate the input embeddings, or feature tokens, for the subsequent MHA layer. Finally, the video representation $\hat{\mathbf{X}}$ is derived by pooling the sequence of feature tokens from the last MHA layer.

The entire process illustrated in Fig. 2 can be formulated as: $\hat{\mathbf{X}} = f_{\Theta_{STCA}}(\mathbf{X})$. Here, $f_{\Theta_{STCA}}$ is a feature extraction function parameterized by Θ_{STCA} that maps the input video $\mathbf{X}$ to the video representation $\hat{\mathbf{X}}$. In this design, the Slow embedding network is vital in determining the local-temporal abstractions necessary for the Transformer to effectively model spatiotemporal interactions.

D. EPISODIC META-LEARNING METHODS FOR TRAINING FEW-SHOT ACTION LEARNERS

In this subsection, we present the foundational elements of common FSL algorithms employed to train few-shot action learners under episodic training process. These algorithms are Reptile [27], Protonet [25], and RelationNet [26]. Besides, we introduce the attention-based RelationNet, which has been specifically tailored for use with our proposed STCA encoder.

For simplicity, we consider a single client in this subsection as the meta-learning algorithm is consistent across all FL clients. Specifically, within this context, this single client possesses a model denoted as $\tilde{\Theta}$ (to distinguish it from the global model Θ and local models $\Theta^{(k)}$). Each training episode consists of a support set $\mathcal{S}$ and an unlabeled query sample $\mathbf{q}$.

1) REPTILE

Reptile [27] is a streamlined meta-learning algorithm designed for efficiency and simplicity. Unlike MAML [7], Reptile does not require complex computations like unrolling computation graphs or calculating second derivatives. Instead, it utilizes straightforward stochastic gradient descent on sampled tasks to iteratively adjust the initial parameters. Reptile begins with an initial model with parameter vector $\tilde{\Theta}$. For each iteration, it performs the following steps to update the weight during the training process:

1. Randomly sample training episode with support set $\mathcal{S}$ and a query sample $\mathbf{q}$.

2. Executes $N (N > 1)$ step of SGD on the model $\tilde{\Theta}$ with support samples from $\mathcal{S}$, resulting in updated parameters W.

3. Updates the initial parameters by moving $\tilde{\Theta}$ towards W using the equation:

$$\tilde{\Theta} \leftarrow \tilde{\Theta} + \epsilon(W - \tilde{\Theta}) \tag{2}$$

where ϵ is the step size for this iteration.

This process continues until the model converges. In the inference phase, a single iteration of the above process is performed with the support set $\mathcal{S}$, and the updated parameters W are used to predict the class probability for query $\mathbf{q}$

2) ProtoNet

ProtoNet [25] is a metric-based classifier that generates prototype representations by averaging feature vectors for each class. This model is trained to assess the similarity between support examples and a query sample. The overall architecture of metric-based meta-learner is illustrated in Figure 3. In a training episode, the meta-learning process involves the following steps:

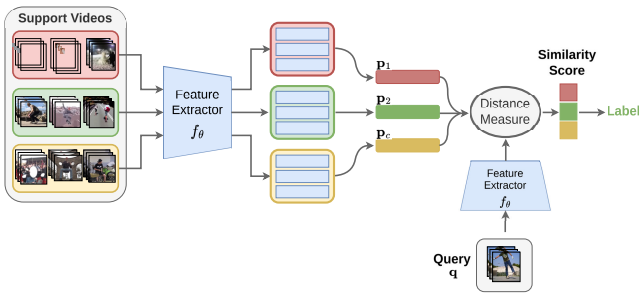

FIGURE 3. Meta-learning process using ProtoNet.

1. Prototype Representation: For each class c in the episode, ProtoNet computes a prototype $\mathbf{p}_c$. This prototype is the mean of the feature representations of all support samples belonging to class c:

$$\mathbf{p}_c = \frac{1}{N_c} \sum_{(\mathbf{x},y)\in\mathcal{S}_c} f_{\tilde{\Theta}}(\mathbf{x})$$
$$\mathcal{S}_c = \{(\mathbf{x}, y) \in \mathcal{S} : y = c\} \tag{3}$$

Here, $\mathcal{S}_c$ represents the subset of support samples for class c, $\mathbf{x}$ is a sample, y is its corresponding class label, and $f_{\tilde{\Theta}}(\cdot)$ is the feature extractor parameterized by $\tilde{\Theta}$.

2. *Similarity Measurement*: The similarity is measured between the query sample $\mathbf{q}$ and each class prototype $\mathbf{p}_c$ using a distance metric, such as the Euclidean distance in this case:

$$d(f_{\tilde{\Theta}}(\mathbf{q}), \mathbf{p}_c) = \|f_{\tilde{\Theta}}(\mathbf{q}) - \mathbf{p}_c\| \tag{4}$$

3. *Class Probability Prediction*: ProtoNet calculates the probability that the query $\mathbf{q}$ belongs to each class c by applying the softmax function to the negative distances between the query features and the prototypes:

$$P(y = c|\mathbf{q}, \mathcal{S}) = \frac{\exp(-d(f_{\tilde{\Theta}}(\mathbf{q}), \mathbf{p}_c))}{\sum_{c'} \exp(-d(f_{\tilde{\Theta}}(\mathbf{q}), \mathbf{p}_{c'}))} \tag{5}$$

During the meta-training phase, ProtoNet uses these class probabilities to train the feature extractor Θ' with a cross-entropy loss on the base classes. In the meta-testing phase, the same process is used to evaluate the model on novel classes, ensuring that it can be generalized to new tasks.

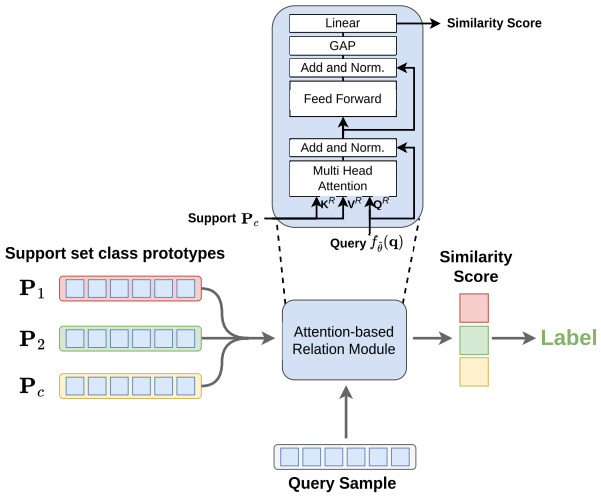

FIGURE 4. Meta-learning process using our attention-based RelationNet.

3) ATTENTION-BASED RELATIONNET

RelationNet [26], similar to ProtoNet, constructs class prototypes by averaging the feature embeddings within each class

in the support set. These prototypes are then concatenated with the query embedding and fed into a relation module, a trainable non-linear operator that computes a similarity score. RelationNet extends ProtoNet with two modules: an embedding module and a relation module. Initially, the embedding module generates representations for both query and training samples, followed by the relation module, which determines whether these embeddings correspond to matching categories. RelationNet was initially devised for few-shot image classification, utilizing 2D CNN blocks in both modules. To adapt it for few-shot action recognition, one can integrate a 3D feature extractor into the embedding module, providing the output feature maps for subsequent convolutional layers in the relation module. However, in the case of our STCA, the CNN-based relation module may not seamlessly align with the Transformer output produced by the embedding module. To tackle this challenge, we introduce the attention-based relation module, tailored for compatibility with embedding modules that generate Transformer feature maps.

As illustrated in Figure 4, the relation module integrates embeddings with a novel cross-attention mechanism to compute similarity scores between two types of inputs, i.e., query samples and support set prototypes. These inputs are processed by trainable neural network layers to produce a final similarity score. The module architecture utilized MHA layers, feed-forward neural networks (FFN), global average pooling (GAP), and linear layers. Here, STCA is modified to output a collection of T token vectors for each video rather than a single feature vector. This adjustment is achieved by extracting the features from layers preceding the pooling operation. The construction of support prototype representation $\mathbf{P}_c$ for class c follows the same methodology outlined in ProtoNet. Both $\mathbf{P}_c \in \mathbb{R}^{T\times d}$ and the query sample representation $f_{\tilde{\theta}}(\mathbf{q}) \in \mathbb{R}^{T\times d}$ contain T token vectors, with each token vectors consist of d dimensions.

The similarity measurement between the query sample and the support prototypes is executed as follows:

1. Linear Mapping: The query representation $f_{\tilde{\theta}}(\mathbf{q})$ is linearly embedded into a matrix $\mathbf{Q}^R$, while $\mathbf{P}_c$ is linearly embedded into matrices $\mathbf{K}^R$ and $\mathbf{V}^R$. This can be formally expressed as: $\mathbf{Q}^R = W_q^R f_\theta(\mathbf{q}), \mathbf{K}^R = W_k^R \mathbf{P}_c, \mathbf{V}^R = W_v^R \mathbf{P}_c$. Here, superscript R denotes the parameter or representation matrices belonging to the relational network.

2. Multi-Head Attention: Using the matrices $\mathbf{Q}^R$, $\mathbf{K}^R$, and $\mathbf{V}^R$, MHA scores are computed by using the formula (1). The output of the MHA layer is then combined with the original query embedding via a shortcut connection and normalized:

$$\text{Attention}^R = \text{LN}(\text{MHA}(\mathbf{Q}^R, \mathbf{K}^R, \mathbf{V}^R) + f_\theta(\mathbf{q})) \quad (6)$$

where $\text{MHA}(\mathbf{Q}^R, \mathbf{K}^R, \mathbf{V}^R)$ represents the output of the MHA. LN is the layer normalization.

3. Feed-Forward Network: The result from the MHA layer is processed through an FFN composed of two 1D convolutional layers. Residual connections and normalization are also applied to the output of these layers:

$$\text{FFN}(\text{Attention}^R) = \text{LN}(\text{Conv1D}_2(\text{ReLU}(\text{Conv1D}_1(\text{Attention}^R))) + \text{Attention}^R) \quad (7)$$

4. Similarity Calculation: The output from the feed-forward network is then pooled and flattened. A final linear layer transforms this output into a similarity score, indicating the match between the query sample and the prototypes:

$$\text{Similarity Score} = w_{\text{out}}(\text{GAP}(\text{FFN}(\text{Attention}^R))) \quad (8)$$

where $w_{\text{out}}(\cdot)$ is the dense layer applied to produce the final similarity score.

The final step involves generating the probability of each class label for the query sample. During the meta-training phase, the loss is propagated backward through both the relation and embedding modules to update parameters and ensure that the model learns to effectively match query samples to class prototypes.

E. FL METHODS FOR AGGREGATING AND UPDATING FEW-SHOT ACTION LEARNERS ACROSS CLIENTS

In this subsection, we present the fundamental principles of FL methods that are needed for aggregating and updating few-shot action learners across multiple clients. We discuss three primary methods: FedAvg, FedProx, and Moon.

1) FEDAVG

FedAvg [72] is a distributed ML algorithm designed for training models across decentralized clients while preserving data privacy. In FedAvg, each device k holds its local dataset $\mathcal{D}_k$, and model parameters Θ are iteratively updated by averaging gradients computed on local data and weighted by the number of samples $|\mathcal{D}_k|$ on each client. At each communication round t, the global model Θ is updated using the following equation:

$$\Theta^{(t+1)} = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \cdot \Theta_k^{(t)} \quad (9)$$

where $\Theta_k^{(t)}$ denotes the model parameters updated on client k at round t, $|\mathcal{D}|$ is the total number of samples across all clients, and K represents the total number of clients participating in the FL process. FedAvg algorithm effectively leverages decentralized data sources for model training while mitigating privacy concerns and communication overhead.

2) FEDPROX

FedProx [4] is an extension of the FedAvg algorithm that incorporates proximal term regularization to improve convergence and robustness in non-convex optimization scenarios. In FedProx, the optimization objective includes a proximal term $R(\Theta)$ added to the standard loss function, which encourages proximity to a reference point Θ^*. At each communication round t, the global model Θ is updated using

the following equation:

$$\Theta^{(t+1)} = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \cdot \left(\Theta_k^{(t)} - \eta \cdot \nabla f_k(\Theta_k^{(t)}) + \lambda \cdot R(\Theta_k^{(t)})\right) \tag{10}$$

where $\Theta_k^{(t)}$ denotes the model parameters updated on client k at round t, $|\mathcal{D}|$ is the total number of samples across all clients, η is the learning rate, λ is the proximal term coefficient, and $f_k(\Theta_k^{(t)})$ is the local loss function on device k. FedProx balances between model accuracy and proximity to the reference point, offering improved convergence properties in FL setups. By using a proximal term, FedProx implicitly addresses issues related to data heterogeneity and non-IID distributions by encouraging model parameters to stay within a certain range defined by the reference point.

3) MOON

MOON [29] is an FL framework tailored for non-IID data. It addresses this challenge by employing contrastive learning principles at the model level. MOON leverages the inherent similarity between the model representations learned by the participating clients. By promoting similar representations for models that excel in comparable data categories, MOON aims to refine the training process and alleviate the adverse effects of non-IID data distributions.

The MOON training process is similar to the FedAvg algorithm. The main difference occurs in the local training stage. Each client has its local dataset, model parameters from previous models Θ_{prev}, and global model parameters Θ_{glob} that are updated on local data and saved as Θ. In each communication round, the model-contrastive loss $\mathcal{L}_{con}$ is calculated as:

$$\mathcal{L}_{con} = -\log \frac{\exp(sim(\Theta, \Theta_{glob})/\tau)}{\exp(sim(\Theta), \Theta_{glob})/\tau) + \exp(sim(\Theta, \Theta_{prev})/\tau)} \tag{11}$$

where τ denotes a temperature parameter. Unlike FedAvg, MOON focuses on aligning the learned representations of the models. By leveraging model representation similarity, MOON mitigates the negative effects of non-IID data.

IV. EXPERIMENTS

In this section, we provide a detailed empirical study of our FedFSLAR++ framework. We begin by introducing benchmark datasets and the experimental settings used for our evaluation. Next, we comprehensively present the results obtained for feature extraction, few-shot action recognition, and FL. Finally, we compare our FedFSLAR++ with the current state-of-the-art methods to validate its robustness.

A. DATASETS

To evaluate the performance of our framework, we perform experiments on four benchmark datasets: Kinetics [2], Something-Something-V2 (SSv2) [73], UCF101 [74], and HMDB51 [75].

1) KINETICS

We utilized a specific subset of Kinetics 400 known as K100, designed for few-shot tasks as presented in CMN [10]. The K100 dataset is composed of 100 classes, each with 100 video samples. These videos encompass everyday activities captured in diverse settings and featuring varying actors and objects. K100 is divided into three non-overlapping subsets: 64 classes for meta-training, 12 classes for meta-validation, and 24 classes for meta-testing.

2) SOMETHING-SOMETHING v2

We employed the Something-Something v2 (SSv2) dataset, which was introduced in OTAM [44]. This dataset consists of 71,718 videos across 100 classes, which are also divided into meta-training, meta-validation, and meta-testing sets. The SSv2 dataset is particularly challenging as it focuses on capturing the dynamics and interactions between objects rather than just their static appearance. This characteristic makes SSv2 an excellent benchmark for testing models on tasks that require an understanding of motion and temporal relationships in video sequences.

3) UCF101

We leveraged UCF101 datasets to evaluate few-shot performance in action recognition. UCF101 contains 13,320 videos distributed over 101 action classes, spanning a wide range of activities such as sports, musical instruments, and human-object interactions. We adopted the dataset split as outlined in ARN [11], this split divided the dataset into a set of 70 classes for training, 10 classes for validation, 21 classes for testing.

4) HMDB51

HMDB51 is another crucial dataset used in our evaluation. HMDB51 is a comprehensive collection of 6,766 video clips categorized into 51 action classes. These classes cover a broad spectrum of human activities, such as facial actions, hand movements, and full-body actions. Following the dataset split strategy from ARN [11], we used a subset of 31 classes for training, 10 classes for validation, 10 classes for testing.

It is important to note that these datasets are widely used for few-shot action recognition. Their inclusion enables a direct and fair comparison of our proposed method with current state-of-the-art approaches. The videos in these datasets were collected from the Internet [2], [74], [75] or crowdsourced platforms [73] in uncontrolled environments [1], presenting real-world challenges. They feature significant variations in camera motion, object appearance, viewpoint, background clutter, and lighting conditions [1]. The video content spans everyday activities across multiple categories, including Body Movement, Human-Object Interaction, Human-Human Interaction, and combinations such as Body-Movement with Object or Human Interaction. These datasets capture practical aspects by incorporating visual appearance and temporal dynamics that challenge models to interpret actions effectively. By using these diverse datasets, we aim to

comprehensively evaluate our FedFSL framework, assessing its ability to handle both appearance-driven and motion-focused tasks. Additionally, the diversity of these datasets allows us to conduct FL experiments that align closely with real-world applications where data variance is high. This helps researchers and practitioners develop models that are adaptive to high data heterogeneity across clients, reflecting the actual challenges of FL deployments.

B. EXPERIMENTAL SETTINGS

1) HYPER-PARAMETERS FOR FEW-SHOT LEARNING

In our experiments, we employ the data augmentation and video preprocessing strategy based on the TSN framework [76]. Specifically, each video is divided into T segments, with one frame sampled from each segment. During training, we randomly sample one frame from each segment. For testing, we select the middle frame from each segment to ensure consistency. We set $T = 16$ frames for MViT [41] and $T = 8$ frames for other tested backbones. Each sampled frame is then resized to 256×256 pixels and randomly cropped to 224×224 during training. In the testing phase, the random crop is replaced by a central crop to standardize evaluation.

To explore the effects of pre-training on FedFSL performance, we conduct our experiments under two distinct settings. In the first setting, we initiate the meta-training process with randomly initialized weights. This approach allows us to evaluate the intrinsic capabilities of different feature backbones and meta-learning algorithms. In the second setting, we utilize open-source backbone models pre-trained on large-scale datasets such as Kinetics-400 (K400) [2], ImageNet [77], and IG-65M [78]. This approach aligns with common practices in the literature [9], [10] and aims to leverage pre-trained networks to boost the FSL performance. To assess the models' performance, we utilized the 5-way N-shot ($N = 1$ or 5) accuracy metric. Across all experiments, the accuracy figures reported are obtained from the mean accuracy over 10,000 randomly sampled episodes from the meta-testing dataset.

a: REPTILE

For the Reptile meta-learning algorithm, meta-training is configured with a 20-shot setup. During this phase, the model is updated using an outer update rate of 0.1. Within each meta-training episode, the inner learning rate is set to 0.02 with SGD optimizer and is scheduled to decrease linearly as training progresses. In the centralized learning scenario, the training consists of 10,000 unique training episodes.

b: METRIC-BASED META LEARNERS

For metric-based approaches (i.e., ProtoNet and RelationNet), we adopt the Adam optimizer for weight updates. When training from scratch with random weights, the learning rate is set to 10^{-4}. For models with pre-trained weights, a reduced learning rate of 10^{-5} is used to fine-tune the model. Training for models initialized with random weights includes 200,000 episodes, while models with pre-trained weights are trained with 100,000 episodes.

2) HYPER-PARAMETERS FOR FEDERATED LEARNING

For federated few-shot settings, we configured a total of 200 training rounds for models with pre-trained weights, and 400 rounds for models initialized with random weights. Each communication round consisted of 400 local training episodes per client. Our solution was tested on setups with 2, 4, 8, 16, and 32 clients, with 4 clients as the default. As a loss function, we used cross-entropy loss. To ensure a fair comparison, these configurations were standardized across all FL algorithms. For the MOON algorithm, we tested only the 5-way 1-shot case due to its intensive memory requirements with additional shots. In contrast, other FL algorithms were evaluated on both 5-way 1-shot and 5-way 5-shot cases. We used accuracy (%) to measure performance. To validate our FL framework's effectiveness under varying data distributions, we performed the IID and non-IID data partitions. In the IID configuration, samples from each class are evenly allocated to all clients. For the non-IID setup, we randomly split the data across clients, varying both the number of action classes and sample counts per client.

C. RESULTS AND ANALYSIS

1) EXPLORATION OF DIFFERENT FEATURE EXTRACTORS

In the first series of experiments, we evaluate the performance of different embedding network types (2D CNN, 3D CNN, Video Transformer, and hybrid) for feature extraction. We report results of 1-shot and 5-shot learning tasks in Tables 3, 4, 5, and 6 on K100, SSv2, UCF101, and HMDB51 datasets, respectively. These experiments were conducted under different FL distribution settings (centralized, IID, and non-IID).

From Table 3, without pre-training, Slow outperforms other CNN feature extractors in K100 due to its strong ability to capture temporal information. Video Transformer models (VTN and MViT) show similar accuracy to CNN models. Combining the strengths of CNNs and Transformer, our STCA stands out with the highest accuracy in the 1-shot and 5-shot learning tasks in all settings. Moreover, when training these models from scratch, CL achieves higher accuracy than FL. This is because the human action diversity of K100 across multiple clients poses challenges to FL performance due to data heterogeneity. In addition, models perform better under IID than non-IID, primarily due to the balanced class distribution in IID settings. When pre-training is considered, we are limited to testing VTN and R(2+1)D models on the K100 dataset because these models offer pre-trained weights sourced from distinct datasets such as ImageNet and IG-65M. In other words, we cannot utilize the pre-trained weights of I3D, R3D, Slow, and MViT models. This limitation stems from these models exclusively open-sourcing weights pre-trained on K400, a superset of K100. Consequently,

employing these pre-trained models would violate the FSL assumption due to class overlap between the source (K400) and target dataset (K100). When used in a centralized setting, both pre-trained VTN and R(2+1)D exhibit comparable performance, significantly surpassing that obtained through random weight initialization. However, VTN, which relies on 2D-CNN for frame-feature extraction and is pre-trained on image dataset, is much less effective in the context of FL compared to 3D-CNN models such as R(2+1)D, which is pre-trained on video action dataset. The accuracy of VTN is reduced when using FL. In contrast, the accuracy of R(2+1)D is greatly improved under FL setting. Therefore, pre-training has a significant impact on boosting FSL performance. Moreover, a 3D feature extractor (e.g., R(2+1)D) pre-trained on video datasets proves far more compatible with FL compared to a 2D feature extractor (e.g., VTN) pre-trained on image datasets.

Table 4 shows the performance on the SSv2 dataset. When initialized with random weights, all models demonstrate poor performance, with CL accuracy nearly corresponding to random guessing on 5-way tasks. The reason is that SSv2 is motion-centric and typically emphasizes spatiotemporal reasoning. These properties make SSv2 more complex and challenging than appearance-based datasets like K100. Thus, meta-training the feature extractor from scratch on this dataset poses a challenge in learning strong representations. Nonetheless, employing FL settings enables us to improve learning performance. Except for the frame-based feature extractor (VTN), spatiotemporal models achieve much higher accuracy than with the centralized setting. Through FL model aggregation in each round, meta-knowledge can be effectively transferred among clients, allowing the model to better escape from the local optima. For SSv2, we can use models pre-trained on K400 since there is no class overlapping between the source and target datasets. Again, the benefit of pre-training for FSL performance is demonstrated clearly in Table 4. With pre-training, MViT and STCA yield better performance than their 3D-CNN counterparts. In this experiment, we also see the impact of different pre-training datasets on R(2+1)D, where K400 is more advantageous than IG-65M. Furthermore, similar to the results on K100, pre-trained feature extractors (excluding VTN) in FL settings outperform those in the centralized setting. Additionally, the results between IID and non-IID are comparable. These outcomes further highlight the advantages of FL for FSL tasks, particularly when integrated with pre-trained feature backbones

We further evaluate the performance of feature extractors on two datasets, UCF101 and HMDB51. Here, UCF101 partially focuses on appearance and scene understanding, similar to K100 [48]. Also, the inter-class variability of UCF101 is higher, making this dataset less challenging. Meanwhile, HMDB51 involves various object interactions and dynamic aspects of actions. Its size is smaller than that of other datasets. In this experiment, we select R(2+1)D and Slow as representatives for CNN-based feature extractors, as they performed the best among other CNNs in previous experiments. These feature extractors are compared with the attention-based model (MViT) and hybrid model (STCA). From Tables 5 and 6, we can observe that, without pre-training, R(2+1)D performs better than MViT and STCA in terms of 1-shot accuracy on both UCF101 and HMDB51 datasets. Compared to R(2+1)D, the 5-shot accuracy of our STCA is higher on HMDB51 but lower on UCF101. Similar to the results shown in Tables 3 and 4, STCA consistently outperforms MViT when training these models from scratch. Nonetheless, with pre-training, MViT and STCA outperform their CNN-based counterparts by a large margin. Again, the self-attention operation used in these pre-trained models demonstrates its effectiveness in learning powerful representations for FSL tasks. The results also show that MViT benefits from pre-training more than STCA. With these datasets, FL clearly exhibits robustness in adapting to new tasks, where the federated accuracy is comparable to or even higher than the centralized accuracy in most cases, even without pre-training. For example, the 1-shot and 5-shot accuracies of STCA in the IID setting are higher than those of the centralized STCA. With the non-IID setting, the accuracy of STCA is slightly decreased due to its higher data heterogeneity among FL clients, but it is still competitive with the centralized accuracy.

Table 7 provides a comparison of various feature extractors in terms of model complexity, i.e., number of parameters, and floating-point operations per second (FLOPs). The I3D model is notable for its minimal complexity, with 28.04M parameters and 57.02G FLOPs, making it highly efficient in terms of computational requirements. R3D, R(2+1)D, and MViT models are significantly more complex. Each of these models has over 30M parameters and needs more than 100G FLOPs to process. The VTN model presents an interesting case with a relatively low number of parameters (24.61M) but a high FLOP count (131.35G). This indicates that VTN, despite having fewer parameters, is computationally demanding. In contrast, the Slow and STCA models find a balance in complexity. The Slow model has 31.63M parameters and requires 83.74G FLOPs, indicating moderate complexity. Meanwhile, the proposed STCA model, although having a higher parameter count at 56.85M, maintains a comparable FLOP requirement at 84.20G. This balance is achieved because the STCA model leverages linear layers that are computationally more efficient than convolutional layers, and its attention mechanisms are also designed to be less heavy on computation due to fewer tokens being processed.

2) DISCUSSION ON FEATURE EXTRACTION

From the results above, we can draw the following conclusions:

- When meta-training feature extractors in a centralized manner with random weight initialization, CNN-based or hybrid models exhibit significantly higher 1-shot accuracy than pure Transformers (e.g., VTN and MViT) due to the limited number of training examples. In 5-shot

settings, while the performance of Transformer models improves, it remains lower than that of other models, particularly our STCA. Pure Transformer layers might be missing some of the useful inductive biases that CNNs and our hybrid model possess [71], necessitating a substantial amount of data to compensate. This lack of inductive biases explains why, in 1-shot tasks without pre-training, MViT and VTN always perform poorly.

- The FSL performance is influenced by the dataset type when incorporating non-pre-trained models with FL. For instance, with a complex appearance-based dataset like K100, federated accuracy is lower than centralized accuracy. However, for UCF101, which has a simpler appearance complexity and higher inter-class variability, federated accuracy can match centralized accuracy. Using our STCA with FL can even surpass the centralized one. Notably, for datasets focusing on motions (e.g., SSv2) or object interactions (e.g. HMDB51), we see a significant boost in FSL performance under FL, clearly demonstrating the great potential of the distributed setting in exploiting spatiotemporal dynamics in videos.
- Under FL settings without pre-training, CNNs and STCA are preferred over pure Transformers like MViT and VTN because the inductive biases of convolution layers enable better preservation of the generalization ability when aggregating few-shot learners.
- Pre-training is especially useful for increasing the FSL performance because meaningful knowledge from external datasets (e.g., K400 or Sport1M) can be transferred to feature extractors. This transferability can better capture motion patterns and adapt to target tasks more effectively. Pre-training tends to benefit attention-based feature extractors (SCTA and MViT) more than CNN-based counterparts. This advantage is due to the attention mechanism's flexibility in modeling relational patterns contextually and higher model capacity, allowing it to generalize better to new action classes.
- Across all datasets and settings, pre-trained MViT and SCTA consistently outperform pre-trained CNN models by a large margin, highlighting the power of self-attention mechanisms in the context of few-shot action recognition. The pre-trained MViT, due to its more sophisticated design, typically delivers better performance than STCA. This superiority is largely attributed to MViT's comprehensive pre-training, which includes initializing the entire model with pre-trained weights. In contrast, STCA uses pre-trained weights only for its convolutional layers, while its multi-head attention layers are initialized randomly.
- Meta-training pre-trained models under FL settings can further boost FSL performance. The federated accuracy exceeds centralized accuracy across all datasets, demonstrating the significant benefits of FL in few-shot action recognition. In this scenario, the meta-knowledge is not only transferred from the external data but also from the local data of multiple FL clients during the aggregation process. In other words, pre-trained models facilitate the transfer of semantic information from external data to subsequent training steps, as well as meta-knowledge exchange among clients. Consequently, the global model can adapt more effectively to new tasks. Essentially, incorporating a strong pre-trained model into FL improves the generalization ability of meta-learners to new clients while addressing data heterogeneity issues, leading to more robust representations.
- When considering the implications of model complexity on performance, it is clear that more complex models, such as MViT and R(2+1)D, can lead to improved performance. However, STCA stands out by providing comparable or even superior results with approximately half the FLOPs of these complex models. This means STCA can perform tasks about twice as fast, which is particularly useful in practical scenarios where computational resources and time are limited. The design of STCA, which uses less computationally demanding linear layers and efficient attention calculations, underscores its capacity to deliver high performance with moderate computational demands, showing its great potential for feature extraction in few-shot action recognition tasks.

3) ABLATION STUDY ON STCA

To understand the optimal configuration for the STCA model, we conducted an ablation study focusing on two key components of this feature extractor: local feature embedding calculation and temporal aggregation. For the local feature embedding, the Slow network was used, and we tested configurations with 4 and 8 input frames. For temporal aggregation, we evaluated three methods: average pooling, max pooling, and multi-head self-attention (MHA) as detailed in Section III-C.1. These components were assessed using three evaluation metrics, including accuracy on SSv2, model size (number of parameters), and the number of FLOPs. The results, presented in Table 8, show that increasing the number of input frames improves the accuracy but nearly doubles the FLOPs, leading to longer processing times. Notably, MHA achieves significantly higher accuracy than average and max poolings due to its superior capability to capture temporal dynamics. Additionally, while MHA requires a higher model size, its number of FLOPs is comparable to those of average and max pooling.

We further examined the impact of MHA-based aggregation by varying the number of layers (L) and attention heads (H), evaluating the model's performance on the SSv2 dataset. Table 9 presents the results, showcasing a trade-off between model complexity and accuracy. We can see that fewer layers and heads yield lower accuracy. Increasing the number of heads improves performance slightly, suggesting that additional attention heads enhance the model's ability to capture complex features. However, further increasing the number of layers leads to performance degradation. For example, (4, 8) configuration significantly increases

TABLE 3. Comparisons of different feature extractors on K100.

		Centralized		IID		non-IID	
Models	**Pre-training**	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
R3D	✗	46.16%	48.94%	41.33%	46.68%	36.24%	45.26%
R(2+1)D	✗	43.58%	57.74%	39.26%	53.46%	37.22%	52.12%
I3D	✗	37.82%	47.78%	36.13%	44.44%	34.18%	42.12%
VTN	✗	44.18%	57.64%	42.18%	53.90%	**40.66%**	**52.34%**
MViT	✗	42.14%	58.24%	36.28%	52.12%	36.44%	52.00%
Slow	✗	**47.88%**	**59.16%**	**42.22%**	**54.28%**	36.46%	44.16%
STCA	✗	**49.16%**	**59.12%**	**42.56%**	**55.12%**	**40.24%**	**54.06%**
R(2+1)D	IG-65M	**69.74%**	**84.64%**	**78.96%**	**90.28%**	**77.32%**	**90.74%**
VTN	ImageNet	**69.54%**	**84.64%**	**69.48%**	**84.18%**	**68.94%**	**82.52%**

Note: Underlined bold and bold denote the best and second best for each setting, respectively.

TABLE 4. Comparisons of different feature extractors on SSv2.

		Centralized		IID		non-IID	
Models	**Pre-training**	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
R3D	✗	**25.58%**	**28.56%**	**32.98%**	**44.04%**	**34.40%**	**44.28%**
R(2+1)D	✗	23.58%	25.48%	31.46%	42.18%	32.44%	42.74%
I3D	✗	24.14%	26.58%	30.58%	40.18%	31.88%	42.30%
VTN	✗	23.44%	26.00%	22.98%	26.12%	22.86%	26.46%
MViT	✗	23.44%	**29.10%**	**33.46%**	42.66%	**34.32%**	**43.02%**
Slow	✗	22.34%	26.42%	23.64%	27.04%	23.14%	26.98%
STCA	✗	**24.48%**	28.16%	28.60%	36.24%	28.02%	35.18%
R3D	K400	30.54%	33.48%	38.4%	49.08%	38.52%	49.78%
R(2+1)D	K400	45.00%	59.02%	45.22%	59.16%	45.98%	59.70%
R(2+1)D	IG-65M	42.52%	53.30%	43.92%	52.10%	44.30%	53.90%
I3D	K400	41.12%	60.82%	43.36%	61.02%	42.26%	61.08%
VTN	ImageNet	44.94%	59.20%	43.12%	55.66%	43.24%	56.62%
MViT	K400	**54.52%**	**70.04%**	**62.14%**	**77.34%**	**61.28%**	**77.82%**
Slow	K400	44.44%	60.66%	45.90%	60.82%	45.48%	61.74%
STCA	K400	**47.22%**	**64.66%**	**52.27%**	**69.37%**	**50.62%**	**66.56%**

Note: Underlined bold and bold denote the best and second best for each setting, respectively.

parameters to 82M with less accuracy, indicating potential overfitting. Meanwhile, the setup (12, 8) escalates the model size to 182.8M with a minimal accuracy improvement, demonstrating that deeper models risk over-parameterization and reduced generalization. Thus, the (2, 8) configuration of the STCA model offers the best balance between accuracy and complexity, making it the optimal choice for our experiments presented in this study.

4) RESOURCE CONSUMPTION AND RUNTIME PERFORMANCE OF STCA

To assess the feasibility of deploying STCA, we conducted experiments to measure its resource consumption and runtime. Table 10 presents the VRAM usage for both inference and training, using the 5-way 1-shot and 5-way 5-shot configurations in FSL. In the 1-shot setting, each episode includes 5 support samples (one per class) and 5 query samples, forming a batch of 10 samples. In the 5-shot setting, each episode consists of 25 support samples (5 per class) and 5 query samples, resulting in a batch of 30 samples. Increasing the number of samples per batch or episode leads to higher memory consumption. The results indicate that during inference, STCA's VRAM usage ranges from 1.9GB to 3.91GB, making it compatible with most devices, including those with limited memory resources. However, training STCA is more memory-intensive, requiring over 9GB of VRAM due to the additional memory demands for gradients and optimizer states. As such, STCA is well-suited for inference on a wide range of devices, while its training is feasible on mid-tier GPU workstations or high-end edge devices.

Table 11 presents runtime performance for the 5-way 1-shot and 5-way 5-shot configurations across four devices equipped with different GPUs: RTX 4090, RTX A4000, RTX 3080, and the edge-oriented AGX Xavier. The RTX GPU versions deliver rapid inference speeds. However, the AGX Xavier, having a unified memory architecture, stands out as a practical option for edge deployment. Due to the VRAM limits of the RTX A4000 (16GB) and RTX 3080 (10GB), training STCA in the 5-shot setting, which requires 30 videos per batch and over 23GB of memory, is not feasible on these devices. In contrast, the AGX Xavier, with its 32GB of unified memory, can handle the 5-way 5-shot configuration, though at a slower rate compared to high-end GPUs. Additionally, we observe that increasing the number of shots leads to a proportional increase in processing time per episode. This delay, despite parallel computation, is likely due to I/O bottlenecks. Thus, lower shot configurations may be preferable in applications requiring real-time responsiveness.

5) QUALITATIVE RESULTS OF DIFFERENT FEATURE EXTRACTORS

In this section, we qualitatively analyze the discriminative power of feature embeddings meta-learned by different

TABLE 5. Comparisons of different feature extractors on UCF101.

		Centralized		IID		non-IID	
Models	**Pre-training**	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
R(2+1)D	✗	**65.96%**	**81.38%**	**68.12%**	**79.00%**	**66.12%**	**80.80%**
MViT	✗	51.92%	72.50%	48.28%	68.1%	47.54%	65.48%
Slow	✗	59.04%	77.22%	61.16%	74.28%	60.32%	76.48%
STCA	✗	**59.88%**	**78.30%**	**63.08%**	**79.16%**	**60.38%**	**76.7%**
R(2+1)D	K400	82.64%	94.5%	83.32%	95.02%	82.6%	94.58%
MViT	K400	**97.26%**	**99.40%**	**97.36%**	**99.6%**	**96.44%**	**99.38%**
Slow	K400	88.24%	96.94%	88.30%	97.58%	88.86%	97.66%
STCA	K400	**92.12%**	**97.54%**	**93.36%**	**97.88%**	**91.98%**	**97.74%**

Note: Underlined bold and bold denote the best and second best for each setting, respectively.

TABLE 6. Comparisons of different feature extractors on HMDB51.

		Centralized		IID		non-IID	
Models	**Pre-training**	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
R(2+1)D	✗	**43.24%**	54.62%	**43.9%**	55.66%	**40.48%**	**52.56%**
MViT	✗	35.42%	42.42%	38.04%	49.48%	34.16%	45.62%
Slow	✗	41.9%	**55.98%**	42.58%	**57.36%**	38.78%	51.38%
STCA	✗	**42.28%**	**57.46%**	**42.94%**	**58.58%**	**39.54%**	**54.30%**
R(2+1)D	K400	54.10%	73.16%	54.9%	73.78%	54.56%	71.74%
MViT	K400	**73.62%**	**87.80%**	**73.0%**	**87.4%**	**71.94%**	**86.88%**
Slow	K400	56.48%	80.64%	58.94%	82.06%	54.36%	80.04%
STCA	K400	**62.84%**	**81.4%**	**63.56%**	**80.5%**	**60.4%**	**80.5%**

Note: Underlined bold and bold denote the best and second best for each setting, respectively.

TABLE 7. Model complexity of different feature extractors.

Method	**#Params**	**#FLOPs**
R3D [22]	33.17M	162.91G
R(2+1)D [22]	31.3M	162.45G
I3D [2]	28.04M	57.02G
VTN [40]	24.61M	131.35G
MViT [24]	36.3M	141.46G
Slow [23]	31.63M	83.74G
STCA	56.85M	84.20 G

TABLE 8. Ablation study on each component of STCA.

3D local feature embedings	Temporal aggregation	Accuracy	#Params	#FLOPs
Slow (8 frames)	Average	44.44%	31.6M	83.74G
Slow (8 frames)	Max	28.64%	31.6M	83.74G
Slow (8 frames)	MHA	47.22%	56.8M	84.2G
Slow (4 frames)	Average	38.4%	31.6M	41.87G
Slow (4 frames)	Max	29.18%	31.6M	41.87G
Slow (4 frames)	MHA	43.94%	56.8M	42.12G

models in a 5-way 1-shot setting. We have chosen three representative models that exemplify different architectural types: CNN, Transformer, and hybrid. Specifically, we selected Slow, MViT, and STCA pre-trained on K400, as they demonstrated superior performance in the previous experiments. We compare t-SNE visualizations [79] of feature embeddings under both centralized learning and non-IID FL using SSv2 and UCF101 datasets. As illustrated in Figure 5, MViT exhibits denser intra-class distributions and better inter-class separation compared to the other models on the SSv2 dataset. Additionally, our STCA model demonstrates a strong discriminative capability, outperforming Slow and proving its effectiveness in learning meaningful representations. A similar conclusion can be drawn from Figure 6 for the UCF101 dataset, where both MViT and STCA achieve superior visualization outcomes. On UCF101, due to the high accuracy obtained by MViT, STCA, and Slow, the intra-class distributions are notably compact, and the inter-class separations are more noticeable. These findings are consistent with the quantitative results presented in Tables 4 and 5, highlighting the performance differences among these models.

Interestingly, under federated settings, features within the same class tend to be more compact, and inter-class features become more distinct compared to those learned under a centralized setting. This trend is particularly evident in the challenging dataset like SSv2. These results further validate the benefit of combining pre-trained feature extractors with FL, showcasing the robustness of our proposed method for few-shot action recognition.

6) EXPLORATION OF FSL METHODS

In this subsection, we analyze the performance of various FSL algorithms. We specifically examine Protonet, RelationNet, and Reptile algorithms in centralized and federated settings. The results for the K100 and SSv2 datasets are presented in Tables 12 and 13, respectively. For the K100 dataset, the results are reported without pre-training, whereas for the SSv2 dataset, we present results with pre-training using the K400 dataset. Additionally, we explore the effectiveness of different relation modules used integrating with our proposed STCA model in Table 14.

7) ANALYSIS OF FSL ALGORITHMS

From Table 12, ProtoNet consistently achieves the best performance on K100 in the centralized setting. However, in the

TABLE 9. Effect of changing the number of layers and attention heads on SSv2.

(L,H)	(1, 2)	(1,4)	(1, 8)	(2, 2)	(2, 4)	(2, 8)	(4, 8)	(12,8)
Number of parameters	41.1M	42.2M	44.3M	50.5M	52.6M	56.8M	82M	182.8M
1-shot accuracy	45.38%	46.72%	46.84%	46.4%	45.62%	47.22%	47.02%	47.7%

TABLE 10. GPU memory consumption of STCA for 5-way 1-shot and 5-way 5-shot configurations.

	1-shot	5-shot
Inference	1.9GB	3.91GB
Training	9.21GB	23.1GB

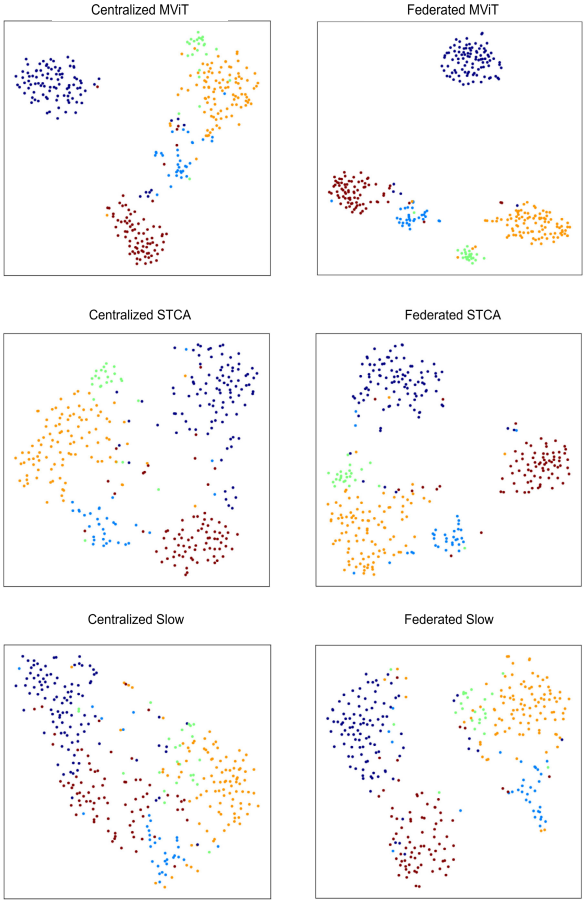

FIGURE 5. t-SNE visualization on the test set of SSv2. Different colors denote different classes.

federated settings (both IID and non-IID), RelationNet outperforms ProtoNet, particularly in the 1-shot scenario, where it maintains a higher accuracy across different settings. This observation suggests that RelationNet's trainable relation module, which utilizes a more complex attention mechanism, offers greater power to the variations encountered in FL. In contrast, ProtoNet's fixed Euclidean distance metric, while simpler and effective in CL, seems to struggle with the aggregation of model updates inherent in FL. Furthermore, we can observe that the difference between the 1-shot and 5-shot accuracies is smaller for RelationNet compared to

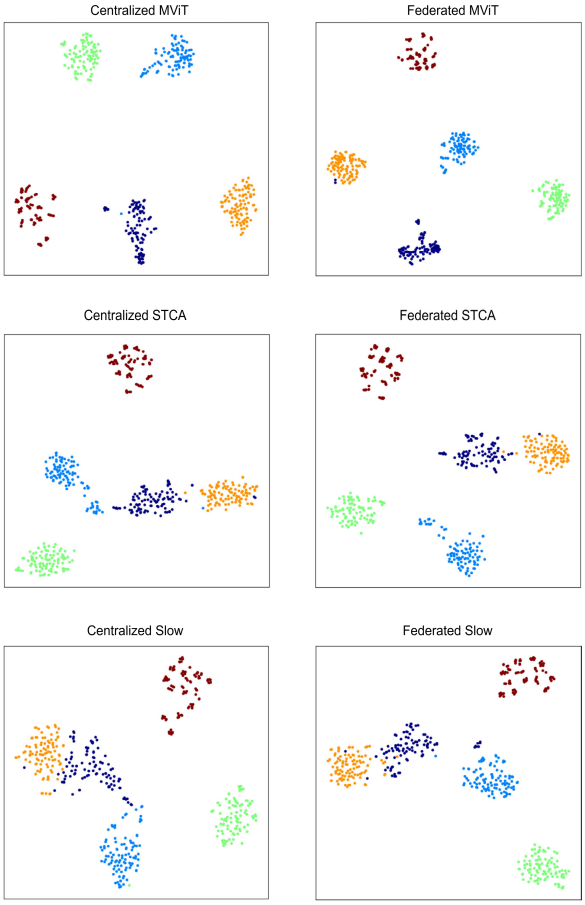

FIGURE 6. t-SNE visualization on the test set of UCF101. Different colors denote different classes.

ProtoNet, implying that RelationNet is better at handling scenarios with extremely limited labeled samples. On the other hand, Reptile exhibits consistently lower accuracy across all settings for the K100 dataset. This significant gap indicates that Reptile, as an optimization-based method, may not be well-suited for handling high-dimensional video data.

From Table 13, ProtoNet with pre-training demonstrates superior performance on SSv2 in all settings, achieving higher accuracy than RelationNet. This could be attributed to the fact that ProtoNet's simpler architecture can better exploit pre-trained weights, which is crucial in leveraging transfer learning from K400. The extra-trainable components of RelationNet, which are not pre-trained, may require more extensive updates to fully align with the pre-trained backbone, potentially undermining the learned representations during fine-tuning. Interestingly, in the SSv2 experiments, FL settings consistently yield better accuracy than CL,

TABLE 11. Runtime (in second) per episode of STCA for 5-way 1-shot and 5-way 5-shot configurations across devices.

	RTX 4090		RTX A4000		RTX 3080		AGX Xavier	
	Inference	Training	Inference	Training	Inference	Training	Inference	Training
1-shot	0.0581	0.1778	0.1231	0.3940	0.0933	0.3025	1.6232	5.6845
5-shot	0.1688	0.5335	0.3569	-	0.2807	-	4.4661	24.637

Note: '-' indicates that the result is not available due to the memory limit of the corresponding device.

TABLE 12. Comparison of different FSL algorithms on K100 dataset.

Distribution	Centralized		IID		non-IID	
Setting	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet	**49.16%**	**59.12%**	42.56%	55.12%	40.24%	54.06%
Reptile	34.9%	42.2%	34.34%	44.12%	35.42%	45.06%
RelationNet	48.60%	56.38%	**48.54%**	**58.58%**	**47.66%**	**57.24%**

TABLE 13. Comparison of different FSL algorithms on SSv2 dataset.

Distribution	Centralized		IID		non-IID	
Setting	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet	**47.22%**	**64.66%**	**52.27%**	**69.37%**	**50.62%**	**66.56%**
Reptile	22.60%	26.36%	24.70%	28.68%	24.76%	28.96%
RelationNet	44.38%	58.50%	45.48%	61.54%	45.0%	60.48%

TABLE 14. Performance of RelationNet when using different relation modules on K100 and SSv2 datasets.

		K100		SSv2	
Embedding	Relation	1-shot	5-shot	1-shot	5-shot
Slow	Conv	44.84%	**59.78%**	43.44%	55.14%
STCA	Linear	20.00%	20.0%	37.06%	44.46%
STCA	MHA	**48.60%**	56.38%	**44.38%**	**58.50%**

particularly for ProtoNet. The consistently poor performance of Reptile, even with pre-training, highlights its limited capability in effectively leveraging pre-trained weights, as evidenced by its near-random accuracy on SSv2.

Overall, ProtoNet's simplicity and consistent high performance across various settings underscore its robust generalization capabilities. Conversely, RelationNet's stable results in FL and its smaller gap between 1-shot and 5-shot performance indicate its potential for specific use cases, particularly in distributed environments. Reptile, however, appears less effective in handling the complexities of the evaluated datasets compared to ProtoNet and RelationNet.

8) ABLATION STUDY ON ATTENTION-BASED RelatioNNet

In the next experiment, we explored the effectiveness of different relation modules. Table 14 summarizes the performance of these modules on K100 and SSv2. The comparison includes STCA with a linear layer, STCA with a MHA-based relation module, and Slow with a convolution-based relation module. The experiments were conducted in a centralized setting, without pre-training for K100, and with pre-training for SSv2.

STCA with a linear layer delivered the lowest accuracy, particularly noticeable in K100, where it presented random guessing behavior. This indicates that linear layers may not capture the complex relationships between examples effectively. On the other hand, integrating a convolution-based relation module with the 3D CNN Slow showed improved performance. Convolution-based relation module performs competently but generally falls short compared to the Transformer-based relation module in most scenarios. Notably, it outperforms MHA from Transformer in the 5-shot setting for K100. This exception may be attributed to K100's focus on appearance features, where convolutional layers are particularly beneficial. Their ability to extract spatial information allows them to compete strongly against Transformer given enough labeled samples. On the other hand, the MHA-based relation module consistently demonstrates its strength in most other settings. For instance, on SSv2, the Transformer-based approach achieves 44.38% accuracy in the 1-shot setting and 58.50% in the 5-shot setting, outperforming both the convolution-based and the linear layer counterparts.

9) EXPLORATION OF FL METHODS

In Tables 15 and 16, we examine different FL methods on the K100 and SSv2 datasets. STCA is primarily used, but results from MViT are also included in some settings. We selected the ProtoNet algorithm as the default for metric-based meta-learning due to its superior performance in previous experiments. Additionally, we included results from the gradient-based algorithm Reptile incorporated into FedAvg, namely FedReptile. For the K100 dataset, raw model weights were used, whereas for the SSv2 dataset, model weights pre-trained on the K400 dataset were utilized.

In most cases, the IID setup shows better performance compared to the non-IID setup. However, the accuracy does not differ significantly. Results in Table 15 further demonstrate the superiority of STCA over MViT when using FedAvg and FedProx without pre-training. Table 16 verifies the benefits of pre-training and different FL algorithms in improving FSL performance. When examining individual FL results, it is clear that FedProx outperforms the others.

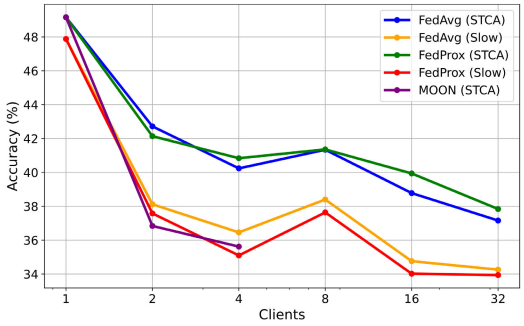

FIGURE 7. Effect of changing the number of clients on K100 dataset.

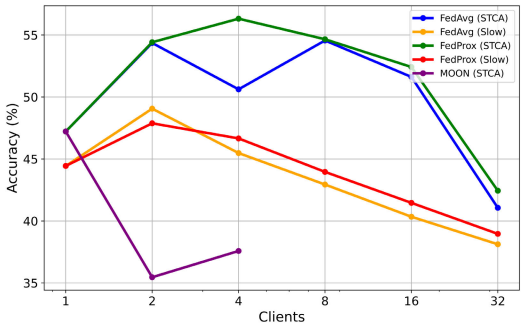

FIGURE 8. Effect of changing the number of clients on SSv2 dataset.

TABLE 15. Performance of different FL methods on K100.

		IID		non-IID	
FL method	Backbone	1-shot	5-shot	1-shot	5-shot
FedAvg	STCA	42.56%	55.12%	40.24%	54.06%
FedAvg	MViT	36.28%	52.12%	36.44%	52.00%
FedProx	STCA	43.56%	55.9%	40.84%	54.8%
FedProx	MViT	37.02%	53.18%	37.28%	53.06%
MOON	STCA	36.0%	-	35.62%	-
FedReptile	STCA	34.34%	44.12	35.42%	45.06

FedAvg results are the second best, while the performance of MOON is poor. Here, the proximal regularization in FedProx encourages the model parameters to remain close to a reference point. This regularization aids in stabilizing the optimization process, especially in non-convex optimization scenarios. It is worth mentioning that MOON is very memory-intensive and expensive to run. Therefore, we could not run 5-way 5-shot cases. Although MOON showed better results on image datasets [29], it struggles with more challenging video datasets. The worst results were obtained using the FedReptile algorithm. On the more challenging SSv2 dataset, FedReptile performs even worse, with accuracy close to random guessing. We suppose that the gradient-based Reptile algorithm performs poorly in capturing complex patterns in action videos.

10) IMPACT OF CHANGING NUMBER OF CLIENTS

In this section, we analyze the effect of changing the number of clients on different FL algorithms in the case of a non-IID 5-way 1-shot with a participation rate of 100%. By default, we used STCA with random weight for the K100 dataset and STCA with pre-trained weights for the SSv2 dataset.

TABLE 16. Performance of different FL methods on SSv2.

		IID		non-IID	
FL method	Backbone	1-shot	5-shot	1-shot	5-shot
FedAvg	STCA	52.27%	69.37%	50.62%	66.56%
FedAvg	MViT	62.14%	77.34%	61.28%	77.82%
FedProx	STCA	58.06%	72.84%	56.32%	72.36%
FedProx	MViT	64.34%	79.12%	63.16%	79.38%
MOON	STCA	34.58%	-	37.58%	-
FedReptile	STCA	24.70%	28.68	24.76%	28.96

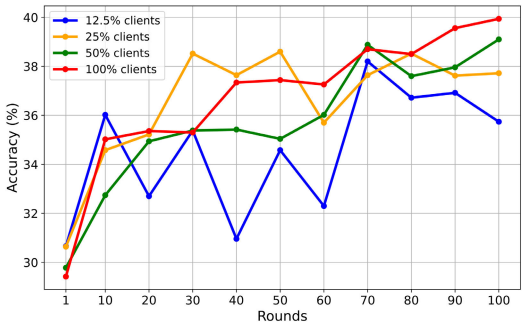

FIGURE 9. Effect of changing the participation rate on K100 dataset.

To further study the effectiveness of STCA, which is an improved version of Slow, we also provide the results for K100 and SSv2 trained on Slow under FedAvg and FedProx algorithms. The K100 dataset results in figure 7, we can observe a steep decline when we increase the number of clients. Because the total number of video samples is fixed, more clients result in fewer training samples per client. This means that when video samples are distributed to more clients, FL struggles to find patterns. Interestingly, with the SSv2 dataset in figure 8, we can observe that increasing the number of clients is even beneficial, especially in the 8 clients setup for the STCA model. Accordingly, for the motion-focused dataset, distributing the data samples sufficiently to each client can help improve the FSL performance. Again, we can see the positive impact of FL with FSL in these more distributed cases. The combination of all these setups gave better results compared to CL. MOON is very memory-consuming, which is the drawback of this model, and we were unable to get the results for more than 4 clients. Also, STCA performs better than the Slow model when changing the number of clients.

11) IMPACT OF CHANGING PARTICIPATION RATE IN EACH ROUND

In this section, we analyze the impact of changing the percentage of participants in each round. By default, we chose 16 clients and tested participation rates of 100%, 50%, 25%, and 12.5%. This means that 16, 8, 4, and 2 clients participate in each training round, respectively. In figure 9, we present the results of K100 for selected clients by running 100 rounds in the FedFSL setup. The feature extractor is STCA. In general, increasing the number of participants is beneficial in later rounds. However, we can observe that even with 2 out of 16 clients, we can achieve 36.02% accuracy, which is only 3% lower than the maximum value of the participation rates.

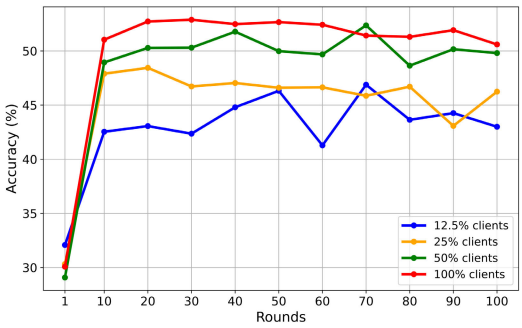

FIGURE 10. Effect of changing the participation rate on SSv2 dataset.

This means that even with partial participation, we can achieve good results in the early rounds. Nevertheless, we can observe that the 2 and 4 client setups did not show a steady growing pattern by fluctuating after 30 rounds. 8 and 16 client setup shows a gradual increase throughout all rounds. This means that increasing the number of participants shows a steady learning curve. Figure 10 shows the results for SSv2. Compared to K100 in Fig. 9, we can observe a better learning curve through all cases except 12.5% case. This result suggests that a higher participation rate can improve the ability to capture temporal dynamics in motion-focused datasets, especially when using pre-trained feature extractors.

12) COMPARISONS WITH STATE-OF-THE-ARTS

In this subsection, we assess the performance of our proposed FedFSLAR++ by benchmarking it against various state-of-the-art (SOTA) methods. The majority of current works employ 2D CNNs, especially ResNet-50, as the feature backbones, which then cooperate with temporal alignment and advanced metric learning to estimate the similarity between query and support samples. Fewer studies have developed the few-shot action recognition framework based on 3D feature extractors. Notably, we compare FedFSLAR++ with twelve SOTA methods (Meta-Baseline [9], Baseline Plus [9], CMN [10], OTAM [44], TRX [45], TA2N [70], [80], MTFAN [81], SloshNet [82], TADRNet [83], HyRSM++ [48], CLIP-FSAR [49]) based on 2D feature extractors and seven SOTA methods (CMOT [52], MASTAF [53], SAFSAR [54], HCR [84], ARN [11], MlSo [85], FedFSLAR [19]) based on 3D feature extractors. The comparison with these methods is conducted in two settings: without and with pre-training, as summarized in Tables 17, 18. In the former setting, we consider the observation from [9] that meta-training few-shot learners from scratch allows for a better understanding of their real generalization performance without violating FSL assumptions. Meanwhile, with the rapid growth of image\video datasets and the availability of numerous pre-trained models, the benefit of pre-training for boosting FSL accuracy is undeniable. Therefore, it is essential to consider both settings for SOTA comparison to gain deeper insights into the effectiveness of our method. We select R(2+1)D, MViT, and STCA as representative feature extractors of 3D CNNs, Video Transformers, and the hybrid model, respectively, and report their best result in each setting from previous experiments. Furthermore, it is noteworthy to emphasize again that our preliminary work [19] and FedFSLAR++ are the first attempts to address few-shot action recognition tasks in FL environments. Other competitors in this subsection were solely developed in a centralized manner.

Results in Table 17 show that, without using pre-trained feature backbones, FedFSLAR++ can perform competitively with existing SOTA methods. Compared to methods based on 2D backbones, FedFSLAR++ demonstrates superior performance on K100. Additionally, the federated accuracy of our FedFSLAR++ with MViT is higher than that of other simple meta-learning approaches (Meta-Baseline [9], CMN [10], and OTAM [44]) on SSv2, further validating the advantage of FL in improving knowledge transferability. However, we achieve inferior results compared to Baseline Plus due to the complex nature of SSv2, which is more favorable to this classifier-based method than to other meta-learning methods. No results have been reported on HMDB51 and UCF101 for 2D backbone-based methods. Table 17 also verifies the effectiveness of 3D feature extractors. For example, on K100, ARN [11] and HCR [84] outperform the 2D backbone-based counterparts by a large margin, highlighting the importance of temporal cues for few-shot action recognition. Compared to FedFSLAR++, ProtoNet [25] with C3D reported in [11] obtains the worse performance, which demonstrates the advantage of R(2+1)D, MViT, and STCA while using the same meta-learning process in the centralized setting. Conversely, ARN, MlSo, and HCR yield better performance on K100, HMDB51, and UCF101. This can be attributed to more complex representations [84], [85] and sophisticated meta-learning strategy [11] proposed by these methods. For example, HCR requires additional computations of human poses and sub-actions to generate hierarchical representation, while MlSo extracts features at multiple levels and scales using second-order pooling. Consequently, the computational complexities of these methods increase significantly.

Table 18 clearly proves the robustness of FedFSLAR++ with pre-training. Specifically, compared to methods utilizing the pre-trained ResNet50 backbone, our FedFSLAR++ significantly outperforms across all datasets and settings. For example, our method with STCA and MViT backbones gains more from pre-training than Baseline Plus. Furthermore, FedFSLAR++ achieves big improvements in 1-shot and 5-shot accuracy over recent advancements in 2D backbone-based methods, including TRX [45], TA2N [70], [80], MTFAN [81], SloshNet [82], TADRNet [83], and HyRSM++ [48], despite employing a simpler meta-training process. Regarding the use of frame-based features, only CLIP-FSAR, which employs a Vision Transformer and the multi-modal knowledge of the CLIP foundation model, outperforms our method on K100. This is because the textual modality of CLIP is advantageous for appearance-based datasets (e.g., K100 and UCF101), whose background

TABLE 17. Comparison with SOTA methods in few-shot action recognition without pre-training.

Method	Distribution	Backbone	K100		SSv2		HMDB51		UCF101	
			1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
Meta-Baseline [9]	Centralized	ResNet-50	42.46%	49.78%	20.85%	21.87%	-	-	-	-
Baseline Plus [9]	Centralized	ResNet-50	46.24%	56.92%	**36.06%**	**48.36%**	-	-	-	-
CMN [10]	Centralized	ResNet-50	40.37%	50.27%	21.1%	23.26%	-	-	-	-
OTAM [44]	Centralized	ResNet-50	44.37%	50.07%	22.75%	23.5%	-	-	-	-
ProtoNet [25]	Centralized	C3D	-	-	-	-	38.05%	53.15%	57.05%	78.25%
ARN [11]	Centralized	C3D	**63.7%**	**82.4%**	-	-	45.5%	**60.6%**	66.3%	83.1%
MlSo [85]	Centralized	C3D	-	-	-	-	**46.7%**	60.3%	**68.2%**	**87.1%**
HCR [84]	Centralized	R(2+1)D	**52.71%**	**68.29%**	-	-	**48.6%**	**63.8%**	**71.8%**	**87.3%**
FedFSLAR [19]	Centralized	Slow	47.88%	59.16%	22.34%	26.42%	41.9%	55.98%	59.04%	77.22%
FedFSLAR [19]	Federated	Slow	47.88%	59.16%	23.64%	27.04%	42.58%	57.36%	61.16%	76.48%
FedFSLAR++	Centralized	R(2+1)D	43.58%	57.74%	23.58%	25.48%	43.24%	54.62%	65.96%	81.38%
FedFSLAR++	Federated	R(2+1)D	39.26%	53.46%	32.44%	42.74%	43.9%	54.7	68.12%	80.8%
FedFSLAR++	Centralized	MViT	42.14%	58.24%	23.44%	29.1%	35.42%	42.42%	51.92%	72.5%
FedFSLAR++	Federated	MViT	36.44%	52.12%	**34.32%**	**43.02%**	38.04%	49.48%	48.28%	68.1%
FedFSLAR++	Centralized	STCA	49.16%	59.12%	24.48%	28.16%	42.28%	57.46%	59.88%	78.3%
FedFSLAR++	Federated	STCA	43.56%	55.9%	28.6%	36.24%	42.94%	58.58%	63.08%	79.16%

Note: Underlined bold and bold denote the best and second best, respectively. '-' indicates that the result is not available in published papers.

TABLE 18. Comparison with SOTA methods in few-shot action recognition with pre-training.

Method	Distribution	Backbone	Pre-training	K100		SSv2		HMDB51		UCF101	
				1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
Meta-Baseline [9]	Centralized	ResNet-50	ImageNet	64.03%	80.43%	37.31%	48.28%	-	-	-	-
Baseline Plus [9]	Centralized	ResNet-50	ImageNet	74.63%	86.62%	46.04%	61.10%	-	-	-	-
CMN [10]	Centralized	ResNet-50	ImageNet	60.5%	78.9%	34.4%	43.8%	-	-	-	-
OTAM [44]	Centralized	ResNet-50	ImageNet	73%	85.8%	42.8%	52.3%	54.5%	68%	79.9%	88.9%
TRX [45]	Centralized	ResNet-50	ImageNet	63.6%	85.9%	42.0%	64.6%	53.1%	75.6%	78.2%	96.1%
TA2N [80]	Centralized	ResNet-50	ImageNet	72.8%	85.8%	47.6%	61%	59.7%	73.9%	81.9%	95.1%
Nguyen et al. [70]	Centralized	ResNet-50	ImageNet	74.3%	87.4%	43.8%	61.7%	59.6%	76.9%	84.9%	95.9%
MTFAN [81]	Centralized	ResNet-50	ImageNet	74.6%	87.4%	45.7%	60.4%	59%	74.6%	84.8%	95.1%
SloshNet [82]	Centralized	ResNet-50	ImageNet	70.4%	87%	46.5%	68.3%	59.4%	77.5%	86%	97.1%
TADRNet [83]	Centralized	ResNet-50	ImageNet	75.6%	87.4%	43%	61.1%	64.3%	78.2%	86.7%	96.4%
HyRSM++ [48]	Centralized	ResNet-50	ImageNet	74%	86.4%	55%	69.8%	61.5%	76.4%	85.8%	95.9%
CLIP-FSAR [49]	Centralized	CLIP-ViT-B	WebImageText	**94.8%**	**95.4%**	62.1%	72.1%	**77.1%**	**87.7%**	96.6%	99%
CMOT [52]	Centralized	C3D	Sports1M	-	-	46.8%	55.9%	66.9%	81.5%	90.4%	95.7%
MASTAF [53]	Centralized	R3D	K700	-	-	50.3%	66.7%	67.9%	81.2%	90.6%	97.6%
MASTAF [53]	Centralized	ViViT	K700	-	-	60.7%	-	69.5%	-	91.6%	-
SAFSAR [54]	Centralized	VideoMAE	K400+Text	-	-	**71.26%**	**76.75%**	**77.38%**	85.63%	**98.3%**	99.54%
HCR [84]	Centralized	R(2+1)D	IG-65M	75.7%	86.4%	-	-	67.5%	79.3%	88.9%	95.7%
FedFSLAR [19]	Centralized	Slow	K400	*	*	44.44%	60.66%	56.48%	80.64%	88.24%	96.94%
FedFSLAR [19]	Federated	Slow	K400	*	*	48.92%	66.06%	58.94%	82.06%	88.86%	97.66%
FedFSLAR++	Centralized	R(2+1)D	K400/IG-65M	69.74%	84.64%	45%	59.02%	54.1%	73.16%	82.64%	94.5%
FedFSLAR++	Federated	R(2+1)D	K400/IG-65M	**78.96%**	**90.28%**	45.98%	59.7%	54.9%	73.78%	83.32%	95.02%
FedFSLAR++	Centralized	MViT	K400	*	*	54.52%	70.04%	73.62%	**87.8%**	97.26%	99.4%
FedFSLAR++	Federated	MViT	K400	*	*	**64.34%**	**79.38%**	73%	87.4%	**97.36%**	**99.6%**
FedFSLAR++	Centralized	STCA	K400	*	*	47.22%	64.66%	62.84%	81.4%	92.12%	97.54%
FedFSLAR++	Federated	STCA	K400	*	*	58.06%	72.84%	63.56%	80.05%	93.36%	97.88%

Note: Best and second-best results are denoted in bold and underlined, respectively. '-' indicates that the result is not available in published papers. '*' means the result is not available due to the overlapping between pre-training and target datasets.

information makes actions easily distinguishable. However, on datasets like SSv2 and HMDB51, which involve more motion information or object interaction, our method yields better performance in most cases. MTFAN and TADRNet are other methods that explore additional data modalities (e.g., text and motion) to produce a multi-modal representation, but their accuracy is lower than ours. This superiority again underscores the effectiveness of spatiotemporal feature backbones, especially when integrated with FL settings. Moreover, our FedFSLAR++ with MViT backbone achieves performance improvements over SOTA methods using different types of 3D feature extractors (i.e., C3D, R3D, R(2+1)D, Slow, VideoMAE, and ViViT). Among 3D backbone-based methods, SAFSAR [54] also exploits text data as an additional modality to generate semantic-aware representation and yield higher 1-shot accuracy than our FedFSLAR++ in SSv2, HMDB51, and UCF101. As noted by [49], incorporating text semantic cues is more useful when the visual information is limited, corresponding to the 1-shot setting. Nonetheless, FedFSLAR++ performs better than SAFSAR in the 5-shot setting when more visual information is available for meta-training. Interestingly, our FedFSLAR++ with STCA surpasses many SOTA methods. For example, it performs better than HCR, which has a more complex representation, on HMDB51 (5-shot) and UCF101(both 1-shot and 5-shot) even though STCA is more computation-efficient than R(2+1)D, as shown in Table 6. On K100, When using the same feature backbone as HCR, i.e., R(2+1)D pre-trained on IG-65M, we gain better performance. Furthermore, we observe that FedFSLAR++ achieves noticeable performance improvement compared to FedFSLAR, thanks to the introduction of attention-based

operations for feature extractors and meta-learning processes. These experimental results demonstrate that our simple yet effective pipeline, which incorporates 3D feature extractors into meta-learners under FL settings, can achieve SOTA performance. In other words, our proposed method provides a strong baseline for future research in few-shot action recognition.

V. CONCLUSION

In this paper, we present an innovative FL framework, FedFSLAR++, for few-shot action recognition. Our proposed framework not only enables privacy protection but also reduces communications and storage costs for FSL tasks under FL settings. These attractive characteristics allow us to effectively learn meta-knowledge from the collective abundance of video data among diverse clients and devices, which strengthens the model's generalization ability to adapt to newly unseen actions. Accordingly, we provide a comprehensive study to benchmark the key components of our framework, including 3D feature extraction, FSL, and FL, in solving few-shot action recognition problems. Results from extensive experiments demonstrate the effectiveness of our FedFSLAR++ by clearly showing the benefits of employing 3D feature extractors and FL settings to achieve superior performance against many state-of-the-art methods.

Considering our benchmarking framework is both simple and effective, we believe it can establish a robust baseline for future research in developing more sophisticated FL and FSL strategies for action recognition. For example, the current FL framework imposes excessive computational demands due to the large size of backbone models, making them difficult to perform training on resource-constrained edge devices. Therefore, to address this issue, one possible direction is to improve the efficiency of our FL framework by leveraging knowledge distillation approaches [60] to transfer the knowledge between lightweight models on edge devices and the central large model on the federated server. In this way, we can exploit the learning capability of the central model, reducing the computational load on edge devices while preserving satisfactory accuracy. Moreover, the improved FL efficiency enables us to verify the effectiveness of our proposed method on very large-scale datasets (e.g., YouTube-8M [86]) across heterogeneous clients, hence helping us better understand the universality and applicability of the FL model. On the other hand, given the vast amount of unlabeled video data, we aim to develop an efficient self-supervised pre-training method within the FL framework, enabling feature backbone networks to learn effective video representations. Furthermore, significant effort is required to create more realistic and complex action datasets, encompassing a wider range of categories and scenarios, to enhance the generalizability and robustness of action models. Besides, recognition performance can be boosted by incorporating additional modalities, such as human skeleton data and text. To achieve this, we seek to develop multimodal fusion techniques within the FedFSL framework.

REFERENCES

[1] Y. Kong and Y. Fu, "Human action recognition and prediction: A survey," *Int. J. Comput. Vis.*, vol. 130, no. 5, pp. 1366–1401, May 2022.
[2] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4724–4733.
[3] G. Yao, T. Lei, and J. Zhong, "A review of convolutional-neural-network-based action recognition," *Pattern Recognit. Lett.*, vol. 118, pp. 14–22, Feb. 2019.
[4] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.
[5] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, Jan. 2021.
[6] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, May 2021.
[7] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2017, pp. 1126–1135.
[8] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: A good embedding is all you need," in *Proc. 16th Eur. Conf. Comput. Vis.* Glasgow, U.K.: Springer, 2020, pp. 266–282.
[9] Z. Zhu, L. Wang, S. Guo, and G. Wu, "A closer look at few-shot video classification: A new baseline and benchmark," 2021, *arXiv:2110.12358*.
[10] L. Zhu and Y. Yang, "Compound memory networks for few-shot video classification," in *Proc. ECCV*, Sep. 2018, pp. 1–16.
[11] H. Zhang, L. Zhang, X. Qi, H. Li, Philip H. S. Torr, and P. Koniusz, "Few-shot action recognition with permutation-invariant attention," in *Proc. 16th Eur. Conf. Comput. Vision (ECCV)*, Glasgow, U.K. Springer, Aug. 2020, pp. 525–542.
[12] C. Feichtenhofer, "X3D: Expanding architectures for efficient video recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 200–210.
[13] C. Fan and J. Huang, "Federated few-shot learning with adversarial learning," in *Proc. 19th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, Oct. 2021, pp. 1–8.
[14] N. A. Tu, T. Huynh-The, K. U. Khan, and Y.-K. Lee, "ML-HDP: A hierarchical Bayesian nonparametric model for recognizing human actions in video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 3, pp. 800–814, Mar. 2019.
[15] X. Sun, S. Yang, and C. Zhao, "Lightweight industrial image classifier based on federated few-shot learning," *IEEE Trans. Ind. Informatvol.*, vol. 19, no. 6, pp. 7367–7376, Jun. 2022.
[16] S. Wang, X. Fu, K. Ding, C. Chen, H. Chen, and J. Li, "Federated few-shot learning," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, New York, NY, USA. Association for Computing Machinery, Aug. 2023, pp. 2374–2385.
[17] D. Cai, S. Wang, Y. Wu, F. X. Lin, and M. Xu, "Federated few-shot learning for mobile NLP," in *Proc. 29th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2023, pp. 1–17.
[18] N. D. Hoang, D. Tran-Anh, M. Luong, C. Tran, and C. Pham, "Federated few-shot learning for cough classification with edge devices," *Appl. Intell.*, vol. 53, no. 23, pp. 28241–28253, Dec. 2023.
[19] N. Anh Tu, A. Abu, N. Aikyn, N. Makhanov, M.-H. Lee, K. Le-Huy, and K.-S. Wong, "FedFSLAR: A federated learning framework for few-shot action recognition," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. Workshops (WACVW)*, Jan. 2024, pp. 270–279.
[20] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9630–9640.
[21] S. X. Hu, D. Li, J. Stühmer, M. Kim, and T. M. Hospedales, "Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 9058–9067.
[22] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6450–6459.
[23] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast networks for video recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6201–6210.

[24] Y. Li, C.-Y. Wu, H. Fan, K. Mangalam, B. Xiong, J. Malik, and C. Feichtenhofer, "MViTv2: Improved multiscale vision transformers for classification and detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 4794–4804.

[25] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. NIPS*, 2017, pp. 4080–4090.

[26] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1199–1208.

[27] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018, *arXiv:1803.02999*.

[28] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, Apr. 2017, pp. 1273–1282.

[29] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10708–10717.

[30] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2009, pp. 124.1–124.11.

[31] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3551–3558.

[32] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4305–4314.

[33] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, vol. 2, Oct. 2003, pp. 1470–1477.

[34] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1725–1732.

[35] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4489–4497.

[36] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2014, pp. 1–9.

[37] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1933–1941.

[38] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE Access*, vol. 6, pp. 1155–1166, 2018.

[39] M. Majd and R. Safabakhsh, "A motion-aware ConvLSTM network for action recognition," *Appl. Intell.*, vol. 49, no. 7, pp. 2515–2521, Jul. 2019.

[40] D. Neimark, O. Bar, M. Zohar, and D. Asselmann, "Video transformer network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 3156–3165.

[41] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, "Multiscale vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6804–6815.

[42] Y. Cao, Q. Tang, F. Yang, X. Su, S. You, X. Lu, and C. Xu, "Re-mine, learn and reason: Exploring the cross-modal semantic correlations for language-guided HOI detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 23435–23446.

[43] M. Wang, J. Xing, J. Mei, Y. Liu, and Y. Jiang, "Actionclip: Adapting language-image pretrained models for video action recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Nov. 21, 2023, doi: 10.1109/TNNLS.2023.3331841.

[44] K. Cao, J. Ji, Z. Cao, C.-Y. Chang, and J. C. Niebles, "Few-shot video classification via temporal alignment," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10615–10624.

[45] T. Perrett, A. Masullo, T. Burghardt, M. Mirmehdi, and D. Damen, "Temporal-relational crosstransformers for few-shot action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 475–484.

[46] S. Zhang, J. Zhou, and X. He, "Learning implicit temporal alignment for few-shot video classification," 2021, *arXiv:2105.04823*.

[47] L. Yang, Y. Huang, and Y. Sato, "Compound prototype matching for few-shot action recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Jan. 2022, pp. 351–368.

[48] X. Wang, S. Zhang, Z. Qing, Z. Zuo, C. Gao, R. Jin, and N. Sang, "HyRSM++: Hybrid relation guided temporal set matching for few-shot action recognition," *Pattern Recognit.*, vol. 147, Mar. 2024, Art. no. 110110.

[49] X. Wang, S. Zhang, J. Cen, C. Gao, Y. Zhang, D. Zhao, and N. Sang, "CLIP-guided prototype modulating for few-shot action recognition," *Int. J. Comput. Vis.*, vol. 132, no. 6, pp. 1899–1912, Jun. 2024.

[50] Y. Xian, B. Korbar, M. Douze, L. Torresani, B. Schiele, and Z. Akata, "Generalized few-shot video classification with video retrieval and feature generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 8949–8961, Dec. 2022.

[51] M. Bishay, G. Zoumpourlis, and I. Patras, "TARN: Temporal attentive relation network for few-shot and zero-shot action recognition," 2019, *arXiv:1907.09021*.

[52] S. Lu, H.-J. Ye, and D.-C. Zhan, "Few-shot action recognition with compromised metric via optimal transport," 2021, *arXiv:2104.03737*.

[53] X. Liu, H. Zhang, H. Pirsiavash, and X. Liu, "MASTAF: A model-agnostic spatio-temporal attention fusion network for few-shot video classification," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2023, pp. 2507–2516.

[54] Y. Tang, B. Béjar, and R. Vidal, "Semantic-aware video representation for few-shot action recognition," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2024, pp. 6444–6454.

[55] Y. Cao, X. Su, Q. Tang, S. You, X. Lu, and C. Xu, "Searching for better spatio-temporal alignment in few-shot action recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 21429–21441.

[56] S. K. Dwivedi, V. Gupta, R. Mitra, S. Ahmed, and A. Jain, "ProtoGAN: Towards few shot learning for action recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1308–1316.

[57] Y. Fu, L. Zhang, J. Wang, Y. Fu, and Y.-G. Jiang, "Depth guided adaptive meta-fusion network for few-shot video recognition," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 1142–1151.

[58] K. Doshi and Y. Yilmaz, "Federated learning-based driver activity recognition for edge devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 3337–3345.

[59] L. Yuan, Y. Ma, L. Su, and Z. Wang, "Peer-to-peer federated continual learning for naturalistic driving action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2023, pp. 5250–5259.

[60] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large CNNs at the edge," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, Jan. 2020, pp. 14068–14080.

[61] A. Psaltis, C. Z. Patrikakis, and P. Daras, "Deep multi-modal representation schemes for federated 3D human action recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 334–352.

[62] J. Guo, H. Liu, S. Sun, T. Guo, M. Zhang, and C. Si, "FSAR: Federated skeleton-based action recognition with adaptive topology structure and knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 10400–10410.

[63] F. Angelini, Z. Fu, Y. Long, L. Shao, and S. Mohsen Naqvi, "2D pose-based real-time human action recognition with occlusion-handling," *IEEE Trans. Multimedia*, vol. 22, no. 6, pp. 1433–1446, Jun. 2020.

[64] Z. Xiao, X. Xu, H. Xing, F. Song, X. Wang, and B. Zhao, "A federated learning system with enhanced feature extraction for human activity recognition," *Knowledge-Based Syst.*, vol. 229, Oct. 2021, Art. no. 107338.

[65] X. Ouyang, Z. Xie, J. Zhou, G. Xing, and J. Huang, "ClusterFL: A clustering-based federated learning system for human activity recognition," *ACM Trans. Sensor Netw.*, vol. 19, no. 1, pp. 1–32, Feb. 2023.

[66] W. Huang, M. Ye, B. Du, and X. Gao, "Few-shot model agnostic federated learning," in *Proc. 30th ACM Int. Conf. Multimedia*, Oct. 2022, pp. 7309–7316.

[67] X. Xu, S. Niu, Z. Wang, D. Li, H. Yang, and W. Du, "Client selection based weighted federated few-shot learning," *Appl. Soft Comput.*, vol. 128, Oct. 2022, Art. no. 109488.

[68] M. Yang, X. Chu, J. Zhu, Y. Xi, S. Niu, and Z. Wang, "Adaptive federated few-shot feature learning with prototype rectification," *Eng. Appl. Artif. Intell.*, vol. 126, Nov. 2023, Art. no. 107125.

[69] J. Chen, J. Tang, and W. Li, "Industrial edge intelligence: Federated-meta learning framework for few-shot fault diagnosis," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3561–3573, Dec. 2023.
[70] K. D. Nguyen, Q.-H. Tran, K. Nguyen, B. Hua, and R. Nguyen, "Inductive and transductive few-shot video classification via appearance and temporal alignments," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Jan. 2022, pp. 471–487.
[71] Z. Dai, H. Liu, Q. V. Le, and M. Tan, "CoAtNet: Marrying convolution and attention for all data sizes," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2021, pp. 3965–3977.
[72] J. Koneč ný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. NIPS Workshop Private Multi-Party Mach. Learn.*, Jan. 2016, pp. 5–10.
[73] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic, "The, 'something something' video database for learning and evaluating visual common sense," in *Proc. ICCV*, 2017, pp. 5842–5850.
[74] K. Soomro, A. Roshan Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.
[75] D. S. Wishart et al., "HMDB: The human metabolome database," *Nucleic Acids Res.*, vol. 35, no. Database, pp. D521–D526, Jan. 2007.
[76] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks for action recognition in videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2740–2755, Nov. 2019.
[77] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[78] D. Ghadiyaram, D. Tran, and D. Mahajan, "Large-scale weakly-supervised pre-training for video action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12038–12047.
[79] L. V. D. Maaten and G. E. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, Jan. 2008.
[80] S. Li, H. Liu, R. Qian, Y. Li, J. See, M. Fei, X. Yu, and W. Lin, "TA^{2}N: Two-stage action alignment network for few-shot action recognition," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, Jun. 2022, pp. 1404–1411.
[81] J. Wu, T. Zhang, Z. Zhang, F. Wu, and Y. Zhang, "Motion-modulated temporal fragment alignment network for few-shot action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 9141–9150.
[82] J. Xing, M. Wang, B. Mu, and Y. Liu, "Revisiting the spatial and temporal modeling for few-shot action recognition," in *Proc. AAAI Conf. Artif. Intell.*, Jan. 2023, pp. 3001–3009.
[83] X. Wang, W. Ye, Z. Qi, G. Wang, J. Wu, Y. Shan, X. Qie, and H. Wang, "Task-aware dual-representation network for few-shot action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 10, Oct. 2023.
[84] C. Li, J. Zhang, S. Wu, X. Jin, and S. Shan, "Hierarchical compositional representations for few-shot action recognition," *Comput. Vis. Image Understand.*, vol. 240, Mar. 2024, Art. no. 103911.
[85] H. Zhang, H. Li, and P. Koniusz, "Multi-level second-order few-shot learning," *IEEE Trans. Multimedia*, vol. 25, p. 1, 2022.
[86] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "YouTube-8M: A large-scale video classification benchmark," 2016, *arXiv:1609.08675*.

NGUYEN ANH TU received the B.S. degree in electrical and electronics engineering from Ho Chi Minh City University of Technology, Vietnam, in 2010, and the Ph.D. degree in computer science and engineering from Kyung Hee University, Republic of Korea, in 2018. From February 2018 to January 2019, he was a Postdoctoral Researcher in data and knowledge engineering at the Department of Computer Science and Engineering, Kyung Hee University. He is currently working as an Assistant Professor with the Department of Computer Science, Nazarbayev University, Kazakhstan. His research interests include computer vision, machine learning, deep learning, human activity analysis, and big data processing.

NARTAY AIKYN received the B.S. degree in computer science and the M.S. degree in data science from Nazarbayev University, Kazakhstan, in 2022 and 2024, respectively. He is currently working as a Research Assistant with the School of Engineering and Digital Sciences, Nazarbayev University. His research interests include deep learning, computer vision, and few-shot learning.

NURSULTAN MAKHANOV received the B.S. degree in information systems from International IT University, Almaty, Kazakhstan, in 2013, and the M.S. degree in cybersecurity in computer science from George Washington University, Washington, DC, USA, in 2017. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Nazarbayev University, Astana, Kazakhstan. His research interests include federated learning, few-shot learning, image classification, computer vision, and deep learning.

ASSANALI ABU received the B.S. degree in computer science from Nazarbayev University, Astana, Kazakhstan, in 2022, and the M.S. degree in data science from the Department of Computer Science, Nazarbayev University, in 2024. His research interests include computer vision, action recognition, and federated learning.

KOK-SENG WONG (Member, IEEE) received the degree in computer science (software engineering) from the University of Malaya, Malaysia, in 2002, the M.Sc. degree in information technology from Malaysia University of Science and Technology (in collaboration with MIT), in 2004, and the Ph.D. degree from Soongsil University, South Korea, in 2012. He is currently working as an Associate Professor with the College of Engineering and Computer Science, VinUniversity. He has published high-quality research papers in top conferences, including CVPR, ECCV, ICLR, NeurIPS, and WWW. His research interests include security, data privacy, and trustworthy AI, with a strong emphasis on privacy-preserving frameworks.

MIN-HO LEE received the B.S. degree in computer science from Seokyeong University, Seoul, South Korea, in 2012, and the integrated master's and Ph.D. degree in brain and cognitive engineering from Korea University, Seoul, in 2019. He is currently working as an Assistant Professor with the Department of Computer Science, Nazarbayev University, Astana, Kazakhstan. His current research interests include machine learning, brain computer interfaces, and neurofeedback systems.