IEEE IEEE
Advancing Technology for Humanity

Editors:
Duong Van Hai
Shinobu Hasegawa
Hieu Dinh Vo
Son Nguyen

The 17th Knowledge and Systems Engineering

International Conference

November 2025
Da Lat University, Vietnam

THE 17TH INTERNATIONAL CONFERENCE ON KNOWLEDGE AND SYSTEMS ENGINEERING

KSE 2025

November 2025
Da Lat Univerity, Vietnam

COPYRIGHT

The 17th International Conference on
Knowledge and Sytems Engineering (KSE 2025)

Part number: CFP2503I-ART
ISBN: 979-8-3315-8900-4
Online ISSN: 2694-4804

Scalable Federated Learning for Video Anomaly Detection in Distributed Environments

Nhat-Minh Dang
Faculty of Information Technology
Hung Yen University of Technology and Education
Hungyen, Vietnam
nhatminh@utehy.edu.vn

Thi-Thu-Trang Do
Faculty of Information Technology
Hung Yen University of Technology and Education
Hungyen, Vietnam
trangdtt@utehy.edu.vn

Quoc-Viet Hoang
Faculty of Information Technology
Hung Yen University of Technology and Education
Hungyen, Vietnam
viethqict@utehy.edu.vn

Van-Quyet Nguyen*
Faculty of Information Technology
Hung Yen University of Technology and Education
Hungyen, Vietnam
quyetict@utehy.edu.vn

***Abstract*— Video anomaly detection is crucial for large-scale surveillance safety, but centralized approaches face challenges with data volume, transmission overhead, and training delays. This paper introduces a scalable federated learning approach leveraging Apache Spark to enable efficient model training on large distributed datasets. A lightweight 3D CNN–LSTM model is trained in parallel on multiple clients, and global parameters are aggregated using FedAvg. We conduct experiments on a non-IID indoor activity dataset under different client settings, ranging from 2 to 10, to identify the optimal allocation that maximizes accuracy while minimizing training time. Experimental results show that 6 clients provide the best trade-off, achieving 0.7877 accuracy and 0.7554 F1-score while reducing training time by over 30% compared to 2 clients. When the number of clients exceeds this threshold, the dataset becomes excessively fragmented, diminishing the quality of local training and adversely affecting the accuracy of the aggregated global model. These findings highlight the potential of federated learning, when combined with distributed computing, to enable scalable and time-efficient video anomaly detection in large distributed environments.**

Keywords—Federated Learning, Video Anomaly Detection, Smart Healthcare, Big Data Analytics

I. INTRODUCTION

Video Anomaly Detection (VAD) plays a crucial role in various surveillance applications such as public safety and healthcare. In hospital environments, VAD assists in detecting incidents such as patient falls, dangerous behaviors, or unauthorized access, thereby enhancing safety and enabling timely responses from medical staff. However, practical deployment faces significant computational challenges, as it requires processing continuous video streams from hundreds of cameras with high storage demands and near real-time processing requirements.

Several traditional deep learning approaches rely on centralized training, where video data from multiple cameras is uploaded to a central server to train a unified neural network model [1]. Centralized models based on CNNs, RNNs, transformers, or memory-augmented networks have achieved high accuracy on public datasets [2][3]. Other approaches utilize Graph Convolutional Networks (GCNs) by extracting skeletal information from videos and constructing spatiotemporal graphs to detect abnormal motions [4]. However, these methods typically assume centralized data availability, which becomes infeasible in large-scale distributed systems due to high storage, bandwidth costs, and expensive execution time issues.

To overcome these limitations, Federated Learning (FL) has emerged as a promising alternative. FL allows local training on edge devices and only transmits model weights to a central server for aggregation [5]. FL has been successfully applied in speech recognition, medical imaging, and video anomaly detection. Some studies have proposed unsupervised FL frameworks enabling cameras to collaborate without sharing raw data [6], or integrated FL with semantic knowledge transfer to achieve performance close to that of centralized models [7]. Nevertheless, these works have largely been limited to small-scale setups and lack comprehensive evaluation in large distributed systems.

Several challenges remain when scaling FL. First, scalability: a hospital network may comprise hundreds of continuously operating cameras, making video transmission to a central server impractical. Even FL is subject to performance degradation as the number of devices increases. Second, there is a need for rapid model updates to adapt to new data. Third, data scarcity and non-IID distributions across regions make it difficult to build a generalized model. Therefore, there is a need for an anomaly detection system that is scalable, capable of fast training, and can effectively leverage distributed and heterogeneous data.

This study investigates federated learning for video anomaly detection in distributed, non-IID environments. The system is implemented on Apache Spark cluster as a simulator to evaluate the impact of varying client allocations on model accuracy, execution efficiency, and scalability. FedAvg is employed for its simplicity, ease of deployment, and minimal hyperparameter requirements, reducing experimental complexity. This choice enables a focused analysis of key factors influencing system performance, such as client allocation, computational cost, and scalability, and provides a foundation for future evaluations of more advanced federated learning algorithms in complex real-world scenarios.

The main contributions of this paper are as follows:

- Implementing the FedAvg algorithm in a distributed environment using Apache Spark, aiming toward real-world deployment in hospital surveillance systems.
- Designing and optimizing the internal layers of a 3D CNN–LSTM model for local training on resource-constrained devices, specifically tailored for video data in abnormal behavior detection.
- Analyzing the impact of the number of clients on model performance, accuracy, and processing time,

and proposing optimal configurations to balance computational cost and model efficiency in heterogeneous federated learning settings.

II. Related Work

Most traditional VAD methods rely on centralized training, where all surveillance video data from edge devices is transmitted to a central server for processing and model training. However, in large-scale surveillance systems, such as hospitals with hundreds of continuously operating cameras, this approach becomes impractical due to high transmission costs, significant storage requirements, and latency in processing [8]. Such systems demand fast, scalable, and distributed training capabilities to support near real-time inference.

FL has emerged as an effective solution for machine learning in distributed environments. FL enables edge devices (e.g., cameras or sensor nodes) to train models locally on their private data and only transmit model parameters to a central server for aggregation [9]. This approach not only reduces the bandwidth required for data transmission but also leverages parallelism in the training process, thereby accelerating convergence time.

Recent studies have demonstrated the feasibility of applying FL to VAD tasks. For instance, Doshi et al. [10] proposed FLVAD, a Transformer based architecture integrated with FL to detect anomalies without centralized data access. Al-Lahham et al. [11] introduced CLAP, a fully unsupervised FL framework that utilizes local feedback to train models collaboratively across multiple cameras. In the healthcare domain, Haripriya et al. [12] showed that combining FL with transfer learning can maintain high accuracy while optimizing computational costs and reducing training time.

Nevertheless, scaling FL to real-world distributed systems presents several challenges:

- Scalability and training efficiency: As the number of clients increases, model aggregation time and communication delays grow, leading to slower training processes [13].
- Limited computational resources: Edge devices such as surveillance cameras often lack the capability to handle complex models, necessitating lightweight architectures and efficient local training [14].
- Non-IID data distributions: Variability in data across devices hampers the convergence of a robust global model and requires adaptive aggregation and coordination strategies [9][13].

Therefore, integrating Federated Learning with a distributed computing platform such as Apache Spark is a promising direction toward building scalable and efficient VAD systems. Apache Spark provides robust resource management and task distribution across multiple nodes, enabling large-scale FL deployment while maintaining flexibility and high performance.

III. Methodology

This section presents the proposed methodology of the study. We first describe the overall architecture of the federated learning system. We then outline the design of the local learning model. Finally, we introduce the implementation of the federated training process using Apache Spark.

A. Overall System Architecture

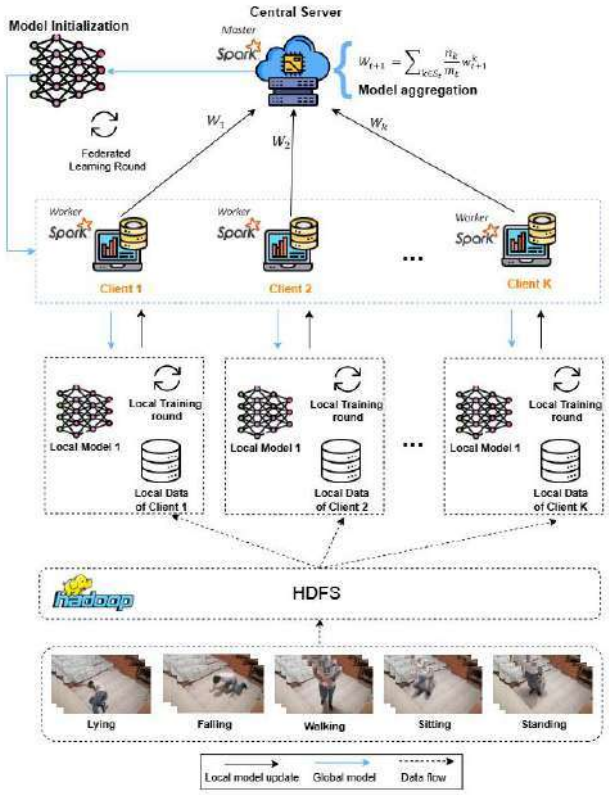

Fig. 1. System Overview

The system is designed to simulate a large-scale distributed deep learning architecture, in which hospital surveillance video data is processed locally at computing nodes following the principles of Federated Learning (FL). The overall system architecture is illustrated in **Fig. 1.**

The first component of the system is the Hadoop Distributed File System (HDFS), which stores preprocessed video segments in the form of short clips. Each clip represents a specific activity, such as falling, lying, walking, sitting, or standing. The data in HDFS is partitioned and evenly distributed across Spark executors to enable parallel processing and ensure load balancing.

Acting as federated clients, the Spark executors perform deep learning model training on their respective data partitions. Each executor may represent a group of cameras, a surveillance zone, or a functional unit within the hospital. The deep learning model deployed at each executor is a lightweight 3D CNN–LSTM architecture, which efficiently captures spatiotemporal features from sequences of video frames.

The Spark driver functions as the central aggregator, responsible for collecting model parameters from the clients using the Federated Averaging (FedAvg) algorithm. After each training round, the global model is updated and broadcast back to the executors for the next federated round.

The proposed architecture enables efficient parallel processing by leveraging multi-core computation at each executor, while preserving data privacy by eliminating the

need to transmit raw video data to a central server. The integration of Apache Spark simplifies scheduling, resource management, and system scalability, supporting both horizontal scaling (increasing the number of executors) and vertical scaling (increasing the training data volume). This makes the system well-suited for large-scale distributed deep learning applications on video data.

B. Design of 3D CNN–LSTM Model Architecture for Video Anomaly Detection

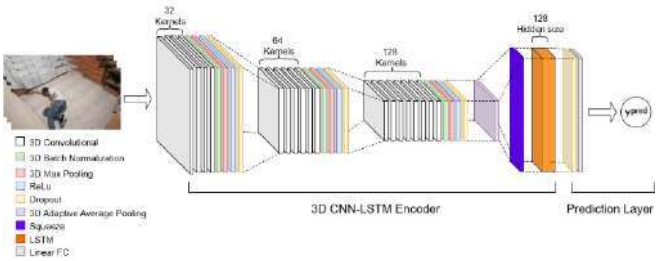

Fig. 2. Proposed 3D CNN-LSTM Model Architecture for Video Anomaly Detection

We propose a model that combines a three-dimensional convolutional neural network (3D CNN) with a Long Short-Term Memory (LSTM) network to jointly capture spatial and temporal features from video sequences for abnormal behavior detection. The model consists of two main components: a 3D CNN–LSTM encoder and a prediction layer. The input is a sequence of video frames with shape (B, T, C, H, W), where B is the batch size, T is the temporal length, C is the number of channels, and H×W is the frame resolution. The input passes through three 3D convolutional blocks with increasing kernel sizes of 32, 64, and 128. Each block includes a 3D convolutional layer with a kernel size of (3×3×3) and padding = 1, which helps preserve the spatial-temporal dimensions. This is followed by a ReLU activation, batch normalization, 3D max pooling (with temporal dimension preserved), and dropout to prevent overfitting.

After feature extraction, the model applies an adaptive average pooling layer to reduce the spatial dimensions to (1×1) while fully preserving the entire temporal dimension T. The resulting feature map is then reshaped to a temporal sequence of shape (B, T, 128) and passed through an LSTM layer with a hidden size of 128. Only the output at the final time step is used as a compact representation of the entire video, which is then passed through a dropout layer and a fully connected linear layer for classification. This architecture is designed to be efficient for local training on resource-constrained devices while maintaining the ability to learn rich spatiotemporal representations from video data.

This design was selected because anomaly detection in video requires capturing both spatial appearance and temporal motion patterns. The 3D CNN–LSTM combination provides a balanced trade-off between expressive capacity and computational feasibility, making it suitable for our controlled simulation of federated learning.

C. Federated Learning Implementation using Spark

To implement the FL process training in a distributed environment, we utilize Apache Spark as a parallel processing platform to distribute data and coordinate the training process across multiple computing nodes. In the Spark architecture, the driver serves as the central coordinator, responsible for task distribution, training management, and aggregation of model parameters after each federated round. The executors function as clients, with each executor performing local model training on its assigned data partition.

The dataset is divided into partitions corresponding to the number of simulated clients. Each executor trains the model on its local data and sends the updated parameters to the driver. The driver aggregates these parameters using the FedAvg algorithm to produce a global model, which is then broadcast back to the executors for the next training round. This process is repeated until convergence or a predefined number of rounds is reached, effectively and flexibly simulating federated learning on a Spark cluster.

IV. Experiments

In this section, we first provides an overview of the video dataset used for the anomaly detection task. We then describes the experimental setup, training parameters, and evaluation metrics used to assess the model's performance in the distributed environment.

A. Dataset

In this study, we use the Indoor Action Dataset released by DaniDeniz et al. [15]. The dataset comprises short video clips capturing daily indoor activities in environments such as kitchens, bedrooms, and living rooms, with a total of 1,228 videos recorded from five different subjects. Although the original dataset contains multiple action labels, we selected only five commonly observed behaviors in patients (see Table I) for model training, aligning with the context of behavior monitoring in hospital settings.

TABLE I. Video Dataset Split

Action Name	Train	Validation	Test	Sum
Falling down	34	11	16	61
Lying on the floor	53	12	36	101
Sitting down	66	22	29	117
Standing up	95	30	41	166
Walking	158	34	90	282
Total	**406**	**109**	**212**	**727**

The videos are preprocessed into sequences of frames with a spatial resolution of 64×64 pixels, ensuring temporal continuity across frames. To address class imbalance and enhance generalization capability, we apply mild data augmentation techniques such as horizontal flipping, color jittering, and geometric transformations. These augmentations increase the diversity of training samples while preserving the behavioral motion characteristics.

B. Experimental Settings

1) Experimental Environment: The experiments were conducted on a Spark cluster consisting of one coordinator node (master) and two compute nodes (workers), each equipped with 6 CPU cores and 8GB of RAM, and running the Ubuntu 20.04 operating system. Apache Spark was configured to manage distributed processing, and the input video data was partitioned and stored using the HDFS, distributed to the corresponding clients (executors acting as clients in the federated learning framework). We run experiments on the datasets with sequence lengths of 20, 15, and 10 frames, respectively. The number of participating

clients was varied (2, 4, 6, 8, and 10), with each client mapped to a dedicated Spark executor to guarantee isolated computation resources. This design enabled a controlled evaluation of how increasing the client scale affects both model performance and system efficiency.

2) Training Configuration: Each experiment consisted of 50 federated rounds. In each round, all participating clients trained their local models for 2 epochs, using a batch size of 4 and a learning rate of 1e-3, optimized with the Adam optimizer. To address class imbalance in the dataset, we employed the Focal Loss function with a focusing parameter gamma = 1.0, and incorporated class weights computed based on label frequency (described in detail in Class Weighting Strategy), enabling the model to better focus on underrepresented classes during training.

3) Class Weighting Strategy: To address the class imbalance problem in the training dataset, we compute class weights based on the frequency of labels in the dataset. Specifically, the raw weight w_i for class i is calculated as:

$$w_i = \frac{N}{C \times n_i} \quad (1)$$

where N is the total number of samples, C is the total number of classes, and n_i is the number of samples belonging to class i. This formula aims to assign higher weights to minority classes and lower weights to majority classes.

To prevent instability caused by excessively large weights during training, these raw weights are softened using the natural logarithm function, resulting in the final weight w'_i calculated as:

$$w'_i = \log(1 + w_i) \quad (2)$$

The softened weights w'_i are then converted to tensors and used in the Focal Loss, helping the model focus better on underrepresented classes while maintaining stability during training.

4) Evaluation Metrics: We use Accuracy, Precision, Recall, and F1-Score to measure the effectiveness of abnormal behavior detection. Simultaneously, time-related metrics such as total training time, average training time per round, model update time, and validation time are recorded to assess the efficiency and scalability of the distributed system.

V. Experimental Results and Analysis

This section presents the experimental results to evaluate the model performance and the execution performance in a distributed federated learning environment.

A. Model Performance Evaluation

Table 2 provides important insights into the influence of client quantity and input frame size on model performance in federated learning systems with non-IID data distribution. The results show that increasing the number of clients from 2 to 6 leads to improved performance, as reflected in rising Accuracy and F1-score values. For example, with 10-frame input sequences, Accuracy increases from 0.7170 (2 clients) to 0.7217 (4 clients), peaking at 0.7877 (6 clients), while F1-score improves from 0.6591 to 0.6599 and then increases to 0.7554. However, when the number of clients increases to 8 and 10, performance begins to decline, with Accuracy dropping to 0.7689 and 0.6085, and F1-score decreasing to 0.7285 and 0.5135, respectively. This pattern suggests that a moderate number of clients can enhance model generalization by improving data diversity and leveraging parallel training. Conversely, an excessive number of clients may fragment the local datasets too severely, reduce the quality of local model updates, and increase communication overhead, all of which negatively affect the global model. These findings highlight the importance of carefully selecting the number of clients to balance computational efficiency and model accuracy in federated learning systems, particularly in real-world applications such as hospital networks where data is inherently distributed.

TABLE II. *Comparison of FL Performance with Different Clients and Data Frame Types*

#Client	Type Frame	Precision	Recall	F1-score	Accuracy
2	10	0.6884	0.6777	0.6591	0.7170
	15	0.7228	0.7779	0.7376	0.7594
	20	0.7170	0.6767	0.6817	0.7170
4	10	0.7035	0.6863	0.6599	0.7217
	15	0.7135	0.7458	0.7224	0.7500
	20	0.7315	0.7791	0.7482	0.7783
6	10	0.7396	0.7789	**0.7554**	**0.7877**
	15	0.7324	0.7497	0.7385	0.7830
	20	0.7780	0.7372	0.7171	0.7830
8	10	0.7290	0.7526	0.7285	0.7689
	15	0.6762	0.6984	0.6742	0.7217
	20	0.7295	0.7372	0.7197	0.7406
10	10	0.5249	0.5383	0.5135	0.6085
	15	0.6974	0.6765	0.6576	0.7264
	20	0.5249	0.5383	0.5135	0.6085

However, when the number of clients exceeds the optimal level (above 6 clients), performance degrades because the data becomes overly fragmented. Each client holds too small a local dataset, weakening the quality of local training and subsequently diminishing the effectiveness of global aggregation via FedAvg. In distributed and heterogeneous data settings, choosing an appropriate number of clients is critical to balancing the benefits of leveraging diverse data and maintaining the stability and efficiency of the federated learning process. The results with short sequence lengths (10 frames) further highlight the importance of preserving sufficient spatiotemporal information for the model to learn and predict accurately in such distributed environments.

B. Execution Performance Evaluation

To evaluate the scalability and time efficiency of the Federated Learning system in a distributed environment, we analyzed the training time, validation time, model update time, and average training time per round across different clients and frame types.

Fig. 3 illustrates the total training time of the FedAvg system under varying client counts and input frame lengths. Increasing the number of clients from 2 to 4 significantly reduces total training time, for example, from 3540.27s to 2436.43s in the 10-frame configuration, due to improved parallelism and more efficient workload distribution. This trend is also observed for 15-frame (5364.33s to 3892.08s) and 20-frame (6790.20s to 4474.50s) inputs.

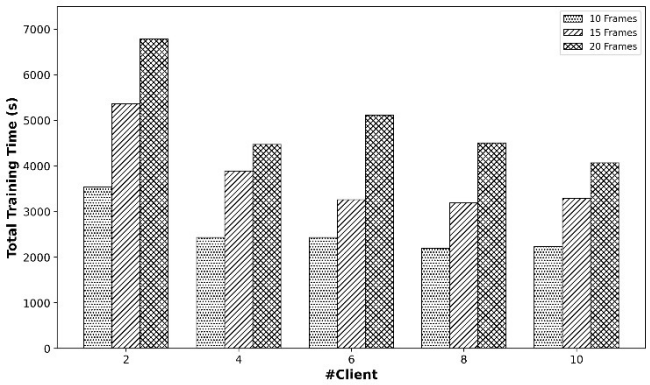

Fig. 3. Comparison of Total Training Time across Different Clients and Frame Types

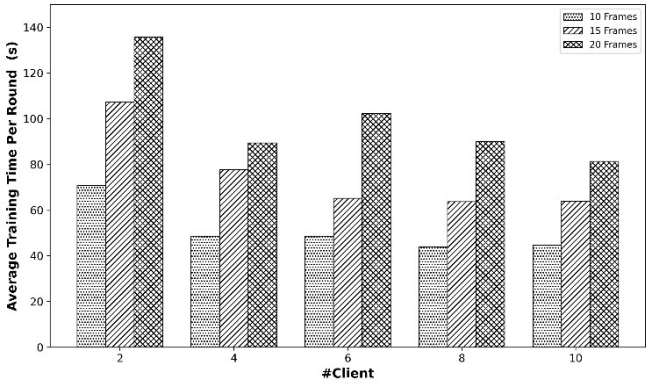

Fig. 4. Comparison of Average Training Time per Round across Different Clients and Frame Types

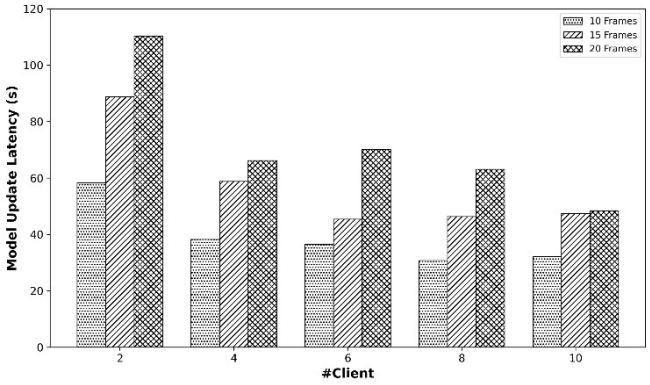

Fig. 5. Comparison of Model Update Time across Different Clients and Frame Types

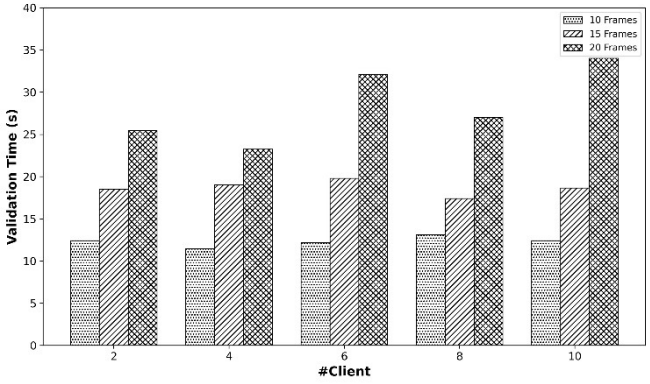

Fig. 6. Comparison of Validation Time across Different Clients and Frame Types

However, when the client count exceeds the optimal threshold of six, the total training time tends to stabilize or slightly increase. For instance, with 20 frames, training time increases from 4474.50s (4 clients) to 5117.84s (6 clients), and fluctuates at 4506.29s (8 clients) and 4069.68s (10 clients). Simultaneously, model accuracy declines, illustrating the trade-off between computational efficiency and model quality due to increased communication and synchronization overhead in large-scale systems.

Fig. 4 presents the average training time per federated round. As the number of clients increases from 2 to 10, round time generally decreases. In the 10-frame setup, it drops from 70.80s (2 clients) to 48.73s (4 clients) and 43.85s (8 clients). However, this metric rises significantly with longer input sequences: for 20-frame inputs, average round time reaches 135.80s (2 clients), 89.49s (4 clients), and 102.36s (6 clients). This reflects increased data complexity and computational cost. When combined with the accuracy trends, these findings highlight the need to carefully select input sequence length and client number to maintain a balance between model performance and computational efficiency.

Fig. 5 depicts the model update time between clients and the central aggregator. As the number of clients increases from 2 to 10, the update time consistently decreases in the 10-frame case from 58.36s (2 clients) to 30.76s (8 clients) and 32.21s (10 clients) due to more efficient synchronization. However, longer sequences still cause moderate increases in latency. For 20-frame inputs, update time peaks at 110.29s (2 clients), decreases to 66.24s (4 clients), but then climbs again to 70.23s (6 clients) and stabilizes around 63.09s (8 clients) and 47.36s (10 clients). This suggests that while scaling helps with synchronization, complex data imposes additional communication load that may impact update consistency and timeliness.

Finally, Fig. 6 shows the validation time after each training round. This metric is mainly influenced by input sequence length rather than client count. For instance, at 10 clients, validation time increases from 12.43s (10 frames) to 18.65s (15 frames) and 34.02s (20 frames). Similar patterns are seen across other client settings. These results indicate that although validation time remains relatively stable across client counts, using longer input sequences to improve accuracy leads to higher validation costs. This emphasizes the importance of balanced system design that considers both model accuracy and computational resource constraints.

Collectively, these findings demonstrate a trade-off between training parallelism and overhead. To ensure both system efficiency and model accuracy, it is essential to balance client allocation and sequence configuration according to available computational resources and task complexity. Such tuning becomes particularly critical in real-time or resource-constrained applications such as smart surveillance or hospital monitoring systems.

VI. Conclusion

This study investigated the integration of federated learning with Apache Spark to enable scalable video anomaly detection in distributed, non-IID environments. Experimental results demonstrate that proper client allocation significantly improves training efficiency while maintaining competitive model performance. Nevertheless, increasing the number of clients beyond an optimal threshold leads to excessive data

fragmentation, weakening local learning quality and degrading the aggregated model. These findings highlight the importance of balancing computational cost, scalability, and accuracy when designing federated learning systems for real-world surveillance scenarios. In the future, we intend to delve into more sophisticated federated learning strategies and adaptive training paradigms to achieve superior robustness, communication efficiency, and generalization capabilities in highly dynamic and heterogeneous distributed settings. We will also evaluate the approach on diverse datasets to validate scalability and robustness across different environments. Additionally, we plan to investigate personalized federated learning methods, optimize communication efficiency, and integrate differential privacy to ensure data confidentiality in real-world deployments.

REFERENCES

[1] G. A. Noghre, "Privacy-preserving Real-world Video Anomaly Detection," 2023 IEEE International Conference on Smart Computing (SMARTCOMP), Nashville, TN, USA, 2023, pp. 235-254, doi: 10.1109/SMARTCOMP58114.2023.00067.

[2] C. Huang, C. Liu, J. Wen, L. Wu, Y. Xu, Q. Jiang, and Y. Wang, "Weakly supervised video anomaly detection via self-guided temporal discriminative transformer," IEEE Transactions on Cybernetics, vol. 54, no. 5, pp. 3197–3210, 2022.

[3] Munyua, John & Wambugu, Geoffrey & Njenga, Stephen. (2021). A Survey of Deep Learning Solutions for Anomaly Detection in Surveillance Videos. International Journal of Computer and Information Technology(2279-0764). 10. 10.24203/ijcit.v10i5.166

[4] Xu, Y.; Huang, B.; Zhou, C.; Wang, H.; Li, X. Video Anomaly Detection with Hyperbolic Graph Embedding and Masked Normalizing Flows. Electronics 2024, 13, 5013. https://doi.org/10.3390/electronics13245013

[5] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... & Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and trends® in machine learning*, *14*(1–2), 1-210.

[6] A. Al-lahham, M. Z. Zaheer, N. Tastan and K. Nandakumar, "Collaborative Learning of Anomalies with Privacy (CLAP) for Unsupervised Video Anomaly Detection: A New Baseline," 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2024, pp. 12416-12425, doi: 10.1109/CVPR52733.2024.01180. keywords: {Training;Privacy;Data privacy;Protocols;Federated learning;Surveillance;Collaboration},

[7] F. Qi, R. Pan, H. Zhang, and C. Xu, "FedVAD: Enhancing Federated Video Anomaly Detection with GPT-Driven Semantic Distillation," in Computer Vision – ECCV 2024, Lecture Notes in Computer Science, vol. 14389, Cham: Springer, 2024, pp. 234–251. doi: 10.1007/978-3-031-73668-1_14.

[8] Liu, Jing & Liu, Yang & Lin, Jieyu & Li, Jielin & Cao, Liang & Sun, Peng & Hu, Bo & Song, Liang & Boukerche, Azzedine & Leung, Victor. (2025). Networking Systems for Video Anomaly Detection: A Tutorial and Survey. ACM Computing Surveys. 57. 10.1145/3729222.

[9] Lu, Zili & Pan, Heng & Dai, Yueyue & Si, Xueming & Zhang, Yan. (2024). Federated Learning With Non-IID Data: A Survey. IEEE Internet of Things Journal. PP. 1-1. 10.1109/JIOT.2024.3376548.

[10] K. Doshi and Y. Yilmaz, "Privacy-Preserving Video Understanding via Transformer-based Federated Learning," 2023 IEEE Conference on Dependable and Secure Computing (DSC), Tampa, FL, USA, 2023, pp. 1-8, doi: 10.1109/DSC61021.2023.10354099.

[11] A. Al-lahham, M. Z. Zaheer, N. Tastan and K. Nandakumar, "Collaborative Learning of Anomalies with Privacy (CLAP) for Unsupervised Video Anomaly Detection: A New Baseline," 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2024, pp. 12416-12425, doi: 10.1109/CVPR52733.2024.01180.

[12] Haripriya, R., Khare, N. & Pandey, M. Privacy-preserving federated learning for collaborative medical data mining in multi-institutional settings. Sci Rep 15, 12482 (2025). https://doi.org/10.1038/s41598-025-97565-4

[13] J. Wang, S. Guo, X. Xie and H. Qi, "Protect Privacy from Gradient Leakage Attack in Federated Learning," IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, United Kingdom, 2022, pp. 580-589, doi: 10.1109/INFOCOM48880.2022.9796841.

[14] Razaque Mughal, Fahad & He, Jingsha & Das, Bhagwan & Dharejo, Fayaz & Zhu, Nafei & Bhatia, Surbhi & Alzahrani, Saeed. (2024). Adaptive federated learning for resource-constrained IoT devices through edge intelligence and multi-edge clustering. Scientific Reports. 14. 10.1038/s41598-024-78239-z.

[15] D. Deniz, E. Ros, E. M. Ortigosa, F. Barranco. "Optimized edge-cloud system for activity monitoring using knowledge distillation" in Electronics, 13 (23), 2024.