FLAMES2GRAPH: AN INTERPRETABLE FEDERATED MULTIVARIATE TIME SERIES CLASSIFICATION FRAMEWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Increasing privacy concerns have led to decentralized and federated machine learning techniques that allow individual clients to consult and train models collaboratively without sharing private information. Some of these applications, such as medical and healthcare, require the final decisions to be interpretable. One common form of data in these applications is multivariate time series, where deep neural networks, especially convolutional neural networks based approaches, have established excellent performance in their classification tasks. However, promising results and performance of deep learning models are a black box, and their decisions cannot always be guaranteed and trusted. While several approaches address the interpretability of deep learning models for multivariate time series data in a centralized environment, less effort has been made in a federated setting. In this work, we introduce FLAMES2Graph, a new horizontal federated learning framework designed to interpret the deep learning decisions of each client. FLAMES2Graph extracts and visualizes those input subsequences that are highly activated by a convolutional neural network. Besides, an evolution graph is created to capture the temporal dependencies between the extracted distinct subsequences. The federated learning clients only share this temporal evolution graph with the centralized server instead of trained model weights to create a global evolution graph. Our extensive experiments on various datasets from well-known multivariate benchmarks indicate that the FLAMES2Graph framework significantly outperforms other state-of-the-art federated methods while keeping privacy and augmenting network decision interpretation.

1 INTRODUCTION

Most machine learning models are trained in a centralized setting and on centrally stored data. This conventional learning approach, in which the clients share their data to a central server, has several drawbacks, such as privacy issues in sensitive applications and limited bandwidth on mobile devices. Moreover, training a single machine learning model on potentially enormous-sized training data on a centralized server or cloud can become absolutely expensive. A practical method of training machine learning models on edge can eliminate the need to train them on the cloud. Federated Learning (FL) Konečnỳ et al. (2016) was introduced to address these issues by only sharing clients' trained model weights with a server in each round of communication. In its original form, the server aggregates the weights using the Federated Averaging Algorithm (FedAvg) algorithm and returns the global model weights to the clients to further use in the next training round. However, training under the FL setting still has several challenges that require further study, such as dealing with non-independent identically distributed (non-IID) data among clients (Younis & Fisichella (2022); Ma et al. (2022)), reducing communication costs (Mills et al. (2019)), handling adversarial attacks (Mothukuri et al. (2021)), and protecting user data privacy (Geyer et al. (2017)). Furthermore, the lack of transparency in the federated learning process diminishes trust and limits its adoption by more clients. It is, therefore, imperative that federated learning approaches be interpretable.

Many recent applications with distributed sensitive data, such as economics forecasting, activity recognition, and healthcare, deal with data collected from sensors (Ullah & Finch (2013); Wang et al. (2016); Sharabiani et al. (2017)). This form of data with simultaneous multiple values is

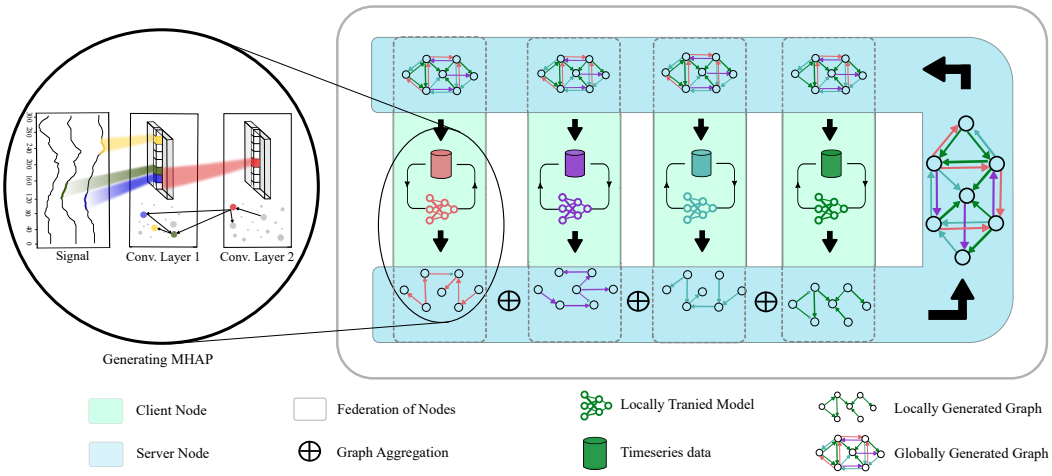


Figure 1: Our proposed FLAMES2Graph framework, an interpretable federated convolutional neural network for multivariate time series data. The figure illustrates four clients that each train a CNN network and extract the highly activated periods (MHAP) from their data to create an MHAP evolution graph in round n. The graph is then shared with the server, and the server aggregates the four clients' graphs into one global graph. Finally, the clients receive the aggregated graph and use it in the following training round $n + 1$.

called multivariate time series (MTS). Various methods have been developed to classify multivariate time series data in a centralized scheme: Distance-based algorithms, such as 1-nearest neighbor and dynamic time warping, have shown reliable performance in case of the correct choice of distance measure for a specific application (Orsenigo & Vercellis (2010); Seto et al. (2015)). On the other hand, feature-based methods heavily rely on hand-crafted features (Ye & Keogh (2009); Xing et al. (2010); D'Urso et al. (2014)). The use of deep learning methods, especially Convolutional Neural Networks (CNN), has shown promising results for time series data (Ismail Fawaz et al. (2019)) and removes the heavy feature engineering requirement. However, despite their outstanding performance, deep learning models' inner functioning and decisions are ambiguous to users and even their designers. Lately, few studies have explored the interpretations of deep networks decisions on time series data in a centralized fashion Cho et al. (2020); Younis et al. (2022).

In this paper, we focus on two major challenges in the federated learning of multivariate time series: The communication efficiency between clients and the server, where the clients usually connect to the FL server over slow connections (Lee et al. (2012)). Another less investigated challenge is designing interpretable strategies for deep neural networks in a federated setup. We design a new graph-based approach called Federated LeArning Multivariate timE Series to Graph (FLAMES2Graph) to address these challenges. To this end, our framework relies on convolutional neural networks trained on multivariate time series data to extract representative patterns that activate neurons in CNNs. We call these representatives Multivariate Highly Activated Period (MHAP) and group their similar patterns into more general representatives via a clustering method. Instead of merely considering their occurrence, we create a graph where its nodes represent the extracted MHAPs cluster medians, and the edges are their occurrence order in an MTS sample. After each training epoch, the client shares its local graph with the server, which represents the influential subsequences found in the local MTS samples. The server then merges the clients' local graphs into a global graph. Figure 1 shows a general overview of our framework. The main contributions of the FLAMES2Graph framework are as follows:

- FLAMES2Graph is the first framework that offers a federated interpretable deep learning solution for the multivariate time series classification problem. It extracts and visualizes the representative patterns of the input data and constructs an MHAP evolution graph that captures the temporal relationship between the extracted representative patterns.
- Instead of sharing the learning weights, the clients only share their generated local graph with the central server, which aggregates those local graphs into a global one to improve the generalization capacity of local clients.

- Through extensive experiments on five selected datasets from Baydogan's archive (Baydogan (2015)), along with HAR (Anguita et al. (2013)) and PAM datasets (Reiss & Stricker (2012)). we verify the FLAMES2Graph framework's performance and interpretability.

2 Related Work

In this section, we review the related work: First, we discuss the existing studies related to interpretable time series deep learning methods. Then, we get an overview of the state-of-the-art federated learning techniques.

2.1 Interpretable Models for time series data

Various approaches have been developed to interpret deep learning models, especially for images and text data. Current research mainly focuses on post-model interpretability strategies, such as Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al. (2016)), Layer Relevance Propagation (LRP) (Montavon et al. (2017)), Class Activation Map (CAM) (Zhou et al. (2016)). CAM was the first post-hoc method that provided interpretation of the predictions by visualizing the region of raw input data that activates the network neurons for a given label (Wang et al. (2017)). Clustered Pattern of Highly Activated Period (CPHAP) (Cho et al. (2020)) is another post-hoc method that extracts the representative input patterns activated by a trained CNN neurons. However, these methods were introduced only for univariate time series data. The multi-VISION framework (Younis et al. (2022)) introduces a method for interpreting multivariate time series data by extracting the input data that highly activate neurons in a CNN. We use a similar approach to extract the representative input data from multivariate time series.

Apart from post-model strategies, Dual-stage Attention-Based Recurrent Neural Network method (DA-RNN) offers an in-model strategy via an input-attention mechanism, captures long-term temporal dependencies, and selects relevant series for model prediction Qin et al. (2017). The Time2Graph framework (Cheng et al. (2020)) is another in-model approach with an intrinsically interpretable strategy that extracts dynamic shapelets from univariate input data through a handcrafted feature engineering procedure and builds an evolution graph based on the temporal sequence of the extracted shapelets. Gated Transformer Networks (GTN) was introduced for multivariate time series by adding gating to transformer networks to explore the GTN attention map (Liu et al. (2021)). An interpretable CNN-based method was developed by Hsieh et al. (2021) for multivariate time series, which identifies both signals and time intervals that determine the classifier's output by adding an attention mechanism to the extracted features. Our approach differs from previous studies since we extract the relevant time series subsequences for multivariate time series and construct an evolution graph of the extracted sequences over time. The benefit of this approach is that no handcraft feature engineering or domain knowledge is required. In addition, our approach can manage horizontally distributed data in a federated learning setup instead of in a central system.

2.2 Federated Learning

Federated Learning is one of the most widely used decentralized machine learning solutions today to train data on clients' edge devices and share only the trained weight (Abdulrahman et al. (2021); Pang et al. (2021)). FedAvg (McMahan et al. (2017)) is the most commonly used optimization algorithm in federated settings. A FedAvg model consists of a global model that is first downloaded and then trained using stochastic gradient descent (SGD) on the local dataset of each participating client. Clients then share their trained model weights with the server, which uses the FedAvg algorithm to average the clients' trained model weights. The communication efficiency of FedAvg is one of its main advantages, as training performance can be maintained while reducing the total number of bytes transferred. However, FedAvg has the disadvantage of degrading convergence when applied to non-IID data (Li et al. (2021)). Different researches were proposed to overcome the FedAvg problem on non-IID data by adding a regularization term to the local optimization or making the data distribution of the clients IID (Li et al. (2020); Karimireddy et al. (2020); Wang et al. (2020)).

In this paper, we present a personalized federated learning approach. Several personalized federated learning methods have been proposed, which can be categorized as transfer learning, meta-learning, personalization layers, and multi-task learning (Kulkarni et al. (2020)). FetchSGD (Rothchild et al.

(2020)) is a federated communication-efficient optimization algorithm that compresses gradients in each communication round with count sketches and sent to a central aggregator. The aggregator maintains a momentum sketch and an error accumulation sketch, and the weight update in each round is computed based on the error accumulation sketch. The proposed technique reduces the communication requirements while fulfilling the quality requirements of federated training. Other research has also addressed representation learning in federated learning, such as (Liang et al. (2020); Collins et al. (2021); Oh et al. (2022)). However, none of the above methods addresses the interpretability of models in the FL setting.

Despite the community's attention to interpretable models in a centralized environment, the problem in a decentralized setting is still understudied. Methods such as LIME (Ribeiro et al. (2016)) cannot be directly applied to the federated setup. SHAP-values (Wang (2019); Zheng et al. (2020)) were introduced for tree-based models and vertically partitioned data; however, in horizontally partitioned data, this method is not applicable. Another federated interpretable method shares the generated dimensionally-reduced intermediate representations of each participating client data Imakura et al. (2021). In this method, neither the models nor the client data are shared. Recently, the iFedAvg method was proposed to detect and solve interoperability problems in federated learning based on tabular datasets (Roschewitz et al. (2021)). None of these methods concentrate on an interpretable federated setting for multivariate time series, the subject of our current research.

3 FLAMES2GRAPH FRAMEWORK

The overall architecture of our proposed framework is shown in Figure. 1. First, each client extracts those subsequences of the multivariate input data that highly activate the neurons in sofar trained CNN (Generate MHAP). Then, each client ("Client Node" in the figure) constructs a graph with nodes representing these time series representative patterns and edges showing their temporal dependency in each sample (Section. 3.1). The federated learning server ("Server Node") then aggregates the clients' local graph into a global graph and shares it with the clients (Section. 3.2). The clients then operate on this global graph in the next round (Section. 3.3).

3.1 FL CLIENT EVOLUTION GRAPH

Cho et al. (2020) and Younis et al. (2022) proposed an approach to extract the input time series periods that highly activated neurons of a trained CNN, with the intuition that decisions of deep neural networks are influenced by their highly activated neurons. We extend their idea to extract each FL client node's multivariate highly activated periods (MHAP) in three steps: First, a convolutional neural network is trained on the multivariate input data. In parallel, an extended set that contains all possible signal combinations of the input sample is created. Each potential combination is created via zero padding of the excluded dimensions. Second, the input set is fed to the trained model to extract MHAPs. An MHAP is extracted by obtaining the trained CNN's Highly Activated Neurons (HAN) and extracting their corresponding signal receptive field from these neurons. Finally, the MHAPs are clustered to merge similar subsequences, and a cluster median is appointed to each cluster.

After getting the MHAPs and cluster centers, each client generates an MHAP evolution graph. The generation of the MHAP evolution graph is inspired by the shapelet evolution graph in Cheng et al. (2020). However, our MHAP evolution graph differs in nature and construction procedure from the shapelet evolution graph. As the MHAPs are extracted separately from each CNN layer, we deal with more than one graph, each representing a CNN layer. The time-dependent graphs are constructed after obtaining the MHAPs from each network layer. These graphs maintain the occurrence order of the MHAPs. Then, all of these graphs are merged into a single time-aware MHAP evolution graph. In the following, the details are discussed.

3.1.1 GENERATE MHAP TIME-AWARE GRAPH

An MHAP graph in each network layer is built as follows: the first MHAP is extracted from the input data and assigned to the corresponding cluster center. This cluster center represents a node n_i in the graph. Then the subsequent MHAP is extracted and assigned to its cluster center n_j, respectively. Now, a directed edge e_{ij} is created between two consecutive nodes, representing the temporal order

Algorithm 1 Generating MHAP Evolution Graph

Input: X_{train}:= Multivariate time series data
Output: MTS Evolution Graph, Cluster Median
1: Model ⟵ Training$_{CNN}(X_{train})$
2: **for** $sample$ `in` X_{train} **do**
3: $signal_{train}$ = InputSet($sample$)
4: **for** l `in Model_layers` **do**
5: **for** c `in` l **do**
6: $MHAP_{list}$ =$extract\_MHAP(signal_{train})$
7: Cluster:= $\{K_{shape}(MHAP_{list})\}$
8: Cluster_Median$_{list}$.$add(Median(Cluster))$
9: **for** l in `Model_layers` **do**
10: Initialize a graph g with n nodes
11: **for** s `in` X_{train} **do**
12: **for** all adjacent MHAP in s **do**
13: Add directed edge $e_{i,i+1}$
14: $Graph_{list}.add(g)$
15: G= $merge(Graph_{list})$
16: **return** Cluster_Median$_{list}$, G

of the MHAPs. The process is repeated for each multivariate sample in the training set to generate a network layer graph G ultimately.

Each node in the resulted graph represents an MHAP of the current layer, e.g., in the second layer, an MHAP is generated from k neurons of the first layer. Note that the size of k depends on the kernel size of the layer. After obtaining the respective graphs of each layer, a merging method compatible with convolutional layers' functioning is applied to the graphs' layers. Thus, we merge each node in the graph with a node in the previous layer. For example, a node n_r in the second layer graph (red node in Generating MHAP in Figure 1) represents three neurons in the first CNN layer (green, blue, and white nodes), two of which are the MHAPs of the first layer, n_b, n_g (the blue and green nodes, respectively), and hence, available in the first layer graph. Accordingly, we create an edge e_{rgb} between the node n_r in the second layer graph and the nodes n_g and n_b in the first layer graph.

Algorithm 1 describes the process of training a model on the FL client node. The algorithm receives the input training data and returns the multivariate time series graph and cluster median in a client. First, the client trains a CNN model for a few epochs and extracts the MHAPs from training data (lines 1-6), and then a k-shape clustering (Paparrizos & Gravano (2015)) is applied to the MHAP list to extract cluster medians (line 7-8). K-shape clustering algorithm considers the unique characteristics of multivariate data, such as phase shift. Similar to k-means, it contains iterative and refining procedures. A shape-based distance measure accounts for phase shift, and a cross-correlation measure identifies centroids. The first step in generating a layer graph is to initialize a graph for each CNN layer with a node size equal to the number of cluster medians for that layer (line 10). Each MHAP is then processed and assigned to its respective clusters, and two adjacent MHAPs are connected by a directed edge (lines 11-13). The generated layer graph is added to the $graph_{list}$ (line 14). In the final stage, the algorithm merges the layers graphs into a single graph (line 15).

3.2 FL Server Graph Aggregation

In each round of communication, the server receives the client's local graphs and cluster medians to aggregate the client's local graphs into a global one. For this purpose, the server first determines the similarities between the nodes of the local graphs using their cluster medians. For example, n_{a1} in the first local graph may be similar to n_{b3} in the second graph. The server first determines the node similarities and then renames the local graphs based on the node similarities. Finally, the graphs are merged by augmenting the edges of all graphs into one. Figure. 2 shows an example of two clients' local graphs with three nodes and the merged graph in the server node. At first, the clients train their model and generate local MHAP graphs (Figure. 2, step 1), then they share their graphs with the server node (Figure. 2, step 2). The server then aggregates the received graphs (Figure. 2,

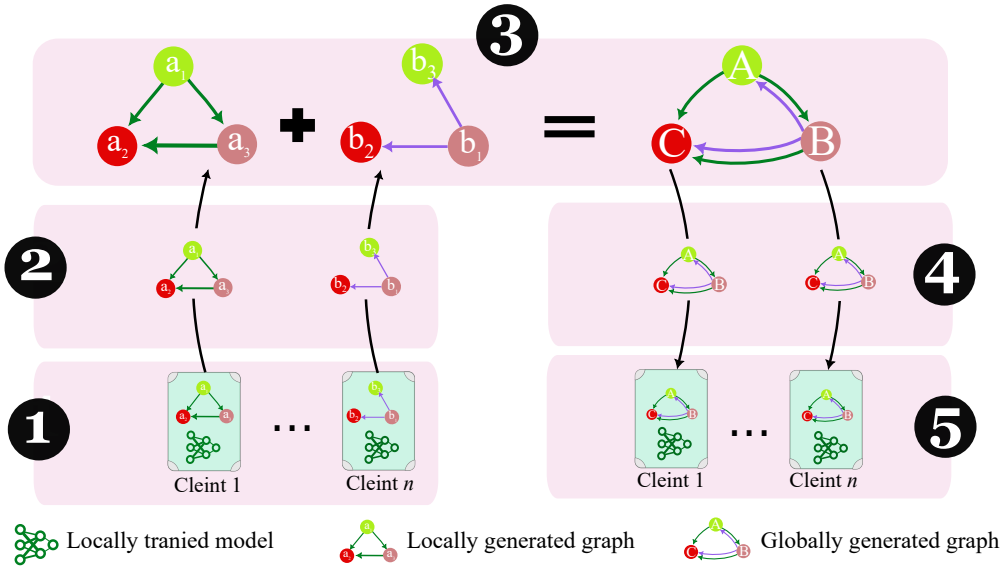


Figure 2: The Client's node in step 1 trains a model and generates the MHAP evolution graphs. In step 2, the clients send the graphs to the server. The server aggregates the local client's graphs in step 3. The global graph forwarded to the clients in step 4. The clients replace their local graphs with the updated global graph in step 5.

step 3) such that the most similar cluster centers are merged, e.g., in Figure. 2, the green node from the first graph, n_{a1}, is similar to n_{b3} in the second graph. The global graph represents this node as n_A with three edges augmented from the first and second local graphs, i.e., e_{AC}, e_{AB}, and e_{BA}. After merging the client's local graphs into a global one, the server returns the global graph to all the clients (Figure. 2, step 4). They will update their local graph into the global one (Figure. 2, step 5) and use it to test their local data in the subsequent communication round.

3.3 FL Client Graph Embedding

After the clients receive the updated global graph from the server, the clients apply the DeepWalk embedding algorithm (Perozzi et al. (2014)) to extract the graph node representation vectors $\Psi \in \mathbb{R}^D$, where D is the embedding size. DeepWalk is a graph neural network algorithm that operates directly on graph structures and applies a random path traversal technique to identify localized structures within a network. DeepWalk identifies latent network patterns through random paths in graphs, which are then learned and encoded by neural networks to generate the final embedding.

The clients use the graph node embedding, Ψ, to convert the test data into segments and make predictions via the graph. For that, each multivariate sample is divided into segments, with the corresponding MHAP assigned to each segment. Each MHAP corresponds to one of the graph nodes n_i, therefore, we retrieve the MHAP representation vector, $\Psi(n_i)$, and sum them over the segment. In the next step, all these vectors are merged into a representation vector Φ for the time series sample. The new vector can then be used as an input to any classifier. Because of the promising performance of the XGBoost classifier (Chen & Guestrin (2016)), the clients use XGBoost to classify the new multivariate representations of time series.

4 Results and Discussion

We report the results of our FLAMES2Graph framework[1] on five well-known time-series benchmarks from the Baydogan archive, and HAR and PAM dataset. (Baydogan (2015)), Our experiments are conducted on a intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz server with 70GB of RAM.

[1]Our code can be found at `https://gitfront.io/r/user-9180183/MCDypNzyNrQf/FLAME2Graph/`

4.1 EXPERIMENTS SETUP

To simulate our federate learning experiments as it is in a real-world scenario, we employ NVIDIA FLARE[2]. We set up a server with four clients for each experiment. Our proposed FLAMES2Graph framework is tested on five datasets from the Baydogan archive (ECG, Wafer, NetFlow, AUSLAN, and UWave), HAR dataset (Anguita et al. (2013)) which records the daily activity of 30 health volunteers, and the PAM dataset (Reiss & Stricker (2012)) that measures eight different daily activities using 17 sensors.

For all datasets, we run 5-fold cross-validation with 80% for training, 10% for validation, and 10% for test data. Each experiment is run for 100 communication rounds. In each round, the client trains the CNN network for 5 epochs. The graph is tested with an embedding of size 100 and a segment length of size 10. For all the datasets, the CNN network has three convolutional blocks with 32, 64, 128 filters, a batch normalization, and a Rectified Linear Unit (ReLU) activation function.

FLAMES2Graph is compared with various benchmark approaches and uses the proposed network architecture and parameters of those benchmark approaches to reproduce similar performance. The benchmark approaches are:

- **FedAvg:** is the first baseline method we compare our model with. FedAvg average the clients trained weights in the central server.
- **FedRep:** learns a shared data representation across clients as well as a unique local head for each client. For every representation update, FedRep performs many local updates based on the low-dimensional local parameters distributed across clients. In every local update, FedRep runs ten epochs of SGD with momentum to train the local head, followed by one or five epochs for the representation.
- **FetchSGD** uses a Count Sketch to compress model updates and benefits from the mergeable nature of sketches to combine model updates from several clients. Since the Count Sketch is linear, momentum and error accumulation can be carried out within the sketch. The goal of FetchSGD framework is to learn a shared data representation across clients as well as a unique local head for each client. In this algorithm, for every update of the representation, many local updates are performed based on the low-dimensional local parameters distributed across clients. FetchSGD framework use a neural network with ReLU activation, a vanilla Count Sketch, non-zero momentum, and momentum factor masking.

4.2 PREDICTION PERFORMANCE

We compare the accuracy, BalanceAccuracy, F1-score, and Precision performance of FLAMES2Graph to several comparison methods on various well-known multivariate time series benchmarks. We run the experiments on four FL clients and report the clients' average performance. For each client, we report its mean value of performance from cross-validation. Table 1 shows that our proposed method outperforms all the other state-of-the-art methods in terms of accuracy, balance accuracy, F1-score, and Precision on UWave, AUSLAN, PAM, and HAR datasets. On ECG and Wafer datasets, FLAMES2Graph improves the accuracy by around 10% and 4%, respectively, compared to the FedAvg and FedRep methods. We observe that FedAvg, FetschSGD, and FedRep do not show promising results when the dataset dimension is large or the time series sequence length is considerable. For example, the Uwave dataset time series length is equal to 315, and the dimensions for AUSLAN are equal to 22. Both show lower accuracy, whereas, in the other datasets with lower dimensions and time series length, the accuracy of FedAvg and FedRep is reasonable. Although the FedAvg and FedRep show satisfactory accuracy results for the Wafer dataset (94%), its balance accuracy, F1-score and precision indicate lower performance. In summary, the experiment results show that our proposed framework, FLAMES2Graph enhances the performance of accuracy for all tested datasets. Also, our method does not merely provide satisfactory classification results for multivariate time series data. It also provides an interpretable network decision, which we will show in a case study in the following section.

[2]FLAIR Documentation:`https://nvflare.readthedocs.io/en/main/flare_overview.html`

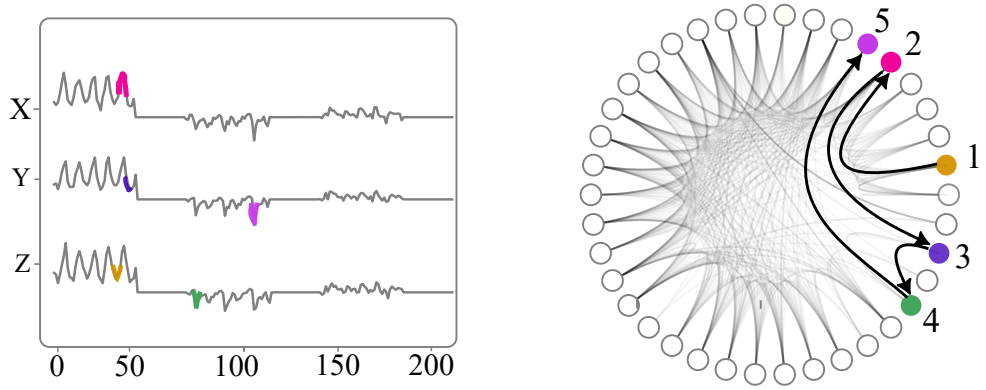
Table 1: The average evaluation metrics for FedAvg, FedRep, FetchSGD, and FLAMES2Graph on different benchmark datasets. The best performance among all models is highlighted.

Dataset	Model	Accuracy	BalanceACC	F1-Score	Precision
ECG	FedAvg	0.83	0.63	0.67	0.64
	FedRep	0.83	0.37	0.58	0.58
	FetchSGD	0.66	0.50	0.52	0.43
	FLAMES2Graph	**0.93**	**0.93**	**0.93**	**0.87**
Wafer	FedAvg	0.94	0.25	0.44	0.44
	FedRep	0.94	0.50	0.95	0.95
	FetchSGD	0.11	0.50	0.02	0.01
	FLAMES2Graph	**0.98**	**0.94**	**0.96**	**1.0**
NetFlow	FedAvg	0.78	0.19	0.39	0.39
	FedRep	0.78	0.44	0.89	0.89
	FetchSGD	0.22	0.50	0.08	0.05
	FLAMES2Graph	**0.88**	**0.89**	**0.91**	**0.93**
AUSLAN	FedAvg	0.10	0.10	0.05	0.05
	FedRep	0.13	0.13	0.11	0.11
	FetchSGD	0.10	0.10	0.05	0.05
	FLAMES2Graph	**0.92**	**0.94**	**0.93**	**1.0**
UWave	FedAvg	0.25	0.12	0.25	0.25
	FedRep	0.30	0.16	0.05	0.05
	FetchSGD	0.12	0.12	0.03	0.01
	FLAMES2Graph	**0.63**	**0.66**	**0.62**	**0.64**
PAM	FedAvg	0.19	0.22	0.22	0.22
	FedRep	0.23	0.12	0.20	0.20
	FetchSGD	0.23	0.12	0.09	0.05
	FLAMES2Graph	**0.65**	**0.60**	**0.57**	**0.58**
HAR	FedAvg	0.23	0.24	0.24	0.24
	FedRep	0.41	0.25	0.23	0.23
	FetchSGD	0.16	0.16	0.05	0.03
	FLAMES2Graph	**0.91**	**0.90**	**0.90**	**0.91**

4.3 CASE STUDY: HAR DATASET

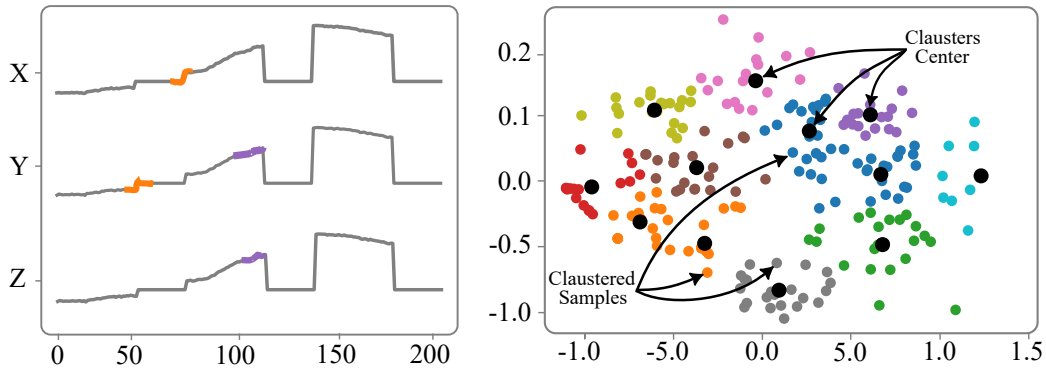
In this section, we take a closer look at the interpretability capacity and the effectiveness of our proposed FLAMES2Graph framework by envisaging a case study from the HAR dataset. Figuer. 3 illustrates an example in a client node with a walking downstairs label and its extracted MHAP Figure. 3 (a). These extracted MHAP are represented as nodes with their temporal transitions in the global MHAP evolution graph obtained from the server Figure. 3 (b). We observe that not only does the client's model performance improve by using the FLAMES2Graph framework, but the client also visualizes the important input subsequences (MHAP) that lead the model to perform well. This graph visualizes the MHAPs and their temporal order in the input data, which is crucial in understanding network decisions on time series data.

Figure. 4 (a) shows another example of the HAR dataset with its respective MHAPs, and Figure. 4 (b) presents the MHAP clusters from the first CNN layer with their cluster medians. In FLAMES2Graph, the federated clients share their local graph and cluster centers with the server; however, this does not reveal any significant information about the client's private data. The length of the centers (the node in the MHAP graph) corresponds to the first kernel size of the network, which is usually not comparable to the original length of the time series, and in this example, equals eight. This means that the length of the center is approximately 3.8% of the total length of the time series sample. Moreover, the MHAP usually does not extract the entire variate, and it only



(a) A Walking Downstairs Sample from HAR Dataset (b) Global MHAP Graph

Figure 3: (a) An example of walking downstairs sample from the HAR dataset with its MHAPs from the first CNN layer, (b) The third round global MHAP evolution graph with the colored nodes highlighting the MHAPs of (a).



(a) A Sample from HAR Dataset (b) Clustered MHAP from HAR Dataset

Figure 4: (a) Another example of walking downstairs sample from the HAR dataset with its MHAP from the first CNN layer, (b) The clustered MHAPs with their centroid (black points).

belongs to one variate of a multivariate sample; hence, these centers usually represent a portion of data dimensions.

5 Conclusion

This work introduces a new personalized federated learning framework designed to interpret deep neural networks for multivariate time series data. We handle interpretability by extracting and visualizing the essential input subsequences that highly activate network neurons in each client and build a temporal evolution graph that captures the time dependencies between these sequences. FLAMES2Graph is novel in its sharing strategy and improves the communication complexity in federated learning by only sharing the evolution graphs. Our extensive experiments indicate that the FLAMES2Graph framework outperforms state-of-the-art federated methods. In the future, we will extend our method by encrypting the graphs and centers sent by the clients to increase the privacy and security of the model.

REFERENCES

Sawsan Abdulrahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497, 2021.

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, and Jorge Luis Reyes Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21th international European symposium on artificial neural networks, computational intelligence and machine learning*, pp. 437–442, 2013.

Mustafa Gokce Baydogan. Multivariate time series classification datasets, 2015. URL `http://www.mustafabaydogan.com/`. [Accessed: 2022-09-23].

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Ziqiang Cheng, Yang Yang, Wei Wang, Wenjie Hu, Yueting Zhuang, and Guojie Song. Time2graph: Revisiting time series modeling with dynamic shapelets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3617–3624, 2020.

Sohee Cho, Ginkyeng Lee, Wonjoon Chang, and Jaesik Choi. Interpretation of deep temporal representations by selective visualization of internally activated nodes. *arXiv preprint arXiv:2004.12538*, 2020.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, pp. 2089–2099. PMLR, 2021.

Pierpaolo D'Urso, Livia De Giovanni, Elizabeth Ann Maharaj, and Riccardo Massari. Wavelet-based self-organizing maps for classifying multivariate time series. *Journal of Chemometrics*, 28 (1):28–51, 2014.

Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

Tsung-Yu Hsieh, Suhang Wang, Yiwei Sun, and Vasant Honavar. Explainable multivariate time series classification: A deep neural network which learns to attend to important variables as well as time intervals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 607–615, 2021.

Akira Imakura, Hiroaki Inaba, Yukihiko Okada, and Tetsuya Sakurai. Interpretable collaborative data analysis on distributed data. *Expert Systems with Applications*, 177:114891, 2021.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *Proc. of the NIPS Workshop on Private Multi-Party Machine Learning*, December 2016.

Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pp. 794–797. IEEE, 2020.

Kyunghan Lee, Joohyun Lee, Yung Yi, Injong Rhee, and Song Chong. Mobile data offloading: How much can wifi deliver? *IEEE/ACM Transactions on networking*, 21(2):536–550, 2012.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. 2021. URL https://arxiv.org/abs/2102.02079.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, volume 2, pp. 429–450, 2020.

Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438*, 2021.

Xiaodong Ma, Jia Zhu, Zhihao Lin, Shanxuan Chen, and Yangjie Qin. A state-of-the-art survey on solving non-iid data in federated learning. *Future Generation Computer Systems*, 135:244–258, 2022. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2022.05.003. URL https://www.sciencedirect.com/science/article/pii/S0167739X22001686.

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

Jed Mills, Jia Hu, and Geyong Min. Communication-efficient federated learning for wireless edge intelligence in iot. *IEEE Internet of Things Journal*, 7(7):5986–5994, 2019.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2020.10.007. URL https://www.sciencedirect.com/science/article/pii/S0167739X20329848.

Jaehoon Oh, SangMook Kim, and Se-Young Yun. Fedbabu: Toward enhanced representation for federated image classification. In *International Conference on Learning Representations*, 2022.

Carlotta Orsenigo and Carlo Vercellis. Combining discrete svm and fixed cardinality warping distances for multivariate time series classification. *Pattern Recognition*, 43(11):3787–3794, 2010.

Junjie Pang, Yan Huang, Zhenzhen Xie, Qilong Han, and Zhipeng Cai. Realizing the heterogeneity: A self-organized federated learning framework for iot. *IEEE Internet of Things Journal*, 8(5): 3088–3098, 2021.

John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1855–1870, 2015.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2627–2633, 2017.

Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th international symposium on wearable computers*, pp. 108–109. IEEE, 2012.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

David Roschewitz, Mary-Anne Hartley, Luca Corinzia, and Martin Jaggi. Ifedavg: Interpretable data-interoperability for federated learning. *arXiv preprint arXiv:2107.06580*, 2021.

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pp. 8253–8265. PMLR, 2020.

Skyler Seto, Wenyu Zhang, and Yichen Zhou. Multivariate time series classification using dynamic time warping template selection for human activity recognition. In *2015 IEEE symposium series on computational intelligence*, pp. 1399–1406. IEEE, 2015.

Anooshiravan Sharabiani, Houshang Darabi, Ashkan Rezaei, Samuel Harford, Hereford Johnson, and Fazle Karim. Efficient classification of long time series by 3-d dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(10):2688–2703, 2017.

Shahid Ullah and Caroline F Finch. Applications of functional data analysis: A systematic review. *BMC medical research methodology*, 13(1):1–12, 2013.

Guan Wang. Interpret federated learning with shapley values. *arXiv preprint arXiv:1905.04519*, 2019.

Jane-Ling Wang, Jeng-Min Chiou, and Hans-Georg Müller. Functional data analysis. *Annual Review of Statistics and Its Application*, 3:257–295, 2016.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7611–7623. Curran Associates, Inc., 2020.

Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585. IEEE, 2017.

Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48, 2010.

Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 947–956, 2009.

Raneen Younis and Marco Fisichella. Fly-smote: Re-balancing the non-iid iot edge devices data in federated learning system. *IEEE Access*, 10:65092–65102, 2022. doi: 10.1109/ACCESS.2022.3184309.

Raneen Younis, Sergej Zerr, and Zahra Ahmadi. Multivariate time series analysis: An interpretable cnn-based model. *in press of The 9th IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2022.

Fanglan Zheng, Kun Li, Jiang Tian, Xiaojia Xiang, et al. A vertical federated learning method for interpretable scorecard and its application in credit scoring. *arXiv preprint arXiv:2009.06218*, 2020.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

Table 2: The statistics of the seven selected datasets from Baydogan's multivariate time series archive.

Dataset	TS-length	Classes	Dimensions	#samples
ECG	152	2	2	200
Wafer	198	2	6	1,194
NetFlow	997	2	4	1,337
AUSLAN	136	95	22	2,565
UWave	315	8	3	4,478
PAM	600	8	17	5,333
HAR	206	6	3	10,299

A Appendix

A.1 Datasets

The Baydogan's archive (Baydogan (2015)), contains 13 multivariate time series classification datasets. In this work, we choose five datasets from this archive (UWave, ECG, AUSLAN, NetFlow, and Wafer).

Human Activity Recognition (HAR) (Anguita et al. (2013)) dataset recorded daily activities from 30 volunteers and produced six different labels of these activities (walking, walking upstairs, walking downstairs, standing, sitting, and lying).

PAMAP2 Physical Activity Monitoring (PAM) (Reiss & Stricker (2012)) dataset recorded 18 different daily activity using 17 sensors, in this work we use 8 activities which contains more than 500 recorded samples.

Table 2 describes the statistics of each dataset.

A.2 Evaluation Metrics

Classification tasks are typically evaluated based on accuracy. In cases where there are fewer minority data points than majority data points, this method may provide acceptable accuracy but may be biased in favor of the majority class. Several evaluation metrics are used to address this issue, including balance accuracy, sensitivity, specificity, Precision, and F1-score . These metrics are defined in the following manner:

$$Sensitivity = \frac{TP}{TP + FN}, \tag{1}$$

$$Specificity = \frac{TN}{FP + TN}, \tag{2}$$

$$BalanceAccuracy = \frac{sensitivity + specificity}{2}. \tag{3}$$

$$Precision = \frac{TP}{TP + FP}, \tag{4}$$

$$F1 - score = 2 * \frac{Precision * sensitivity}{Precision + sensitivity}. \tag{5}$$

where TP is True Positive, TN is True Negative, FP is False Positive, FN is False Negative.