FedTSC: A Secure Federated Learning System for Interpretable Time Series Classification

Zhiyu Liang
Harbin Institute of Technology
Harbin, China
zyliang@hit.edu.cn

Hongzhi Wang*
Harbin Institute of Technology
Harbin, China
wangzh@hit.edu.cn

ABSTRACT

We demonstrate FedTSC, a novel federated learning (FL) system for interpretable time series classification (TSC). FedTSC is an FL-based TSC solution that makes a great balance among security, interpretability, accuracy, and efficiency. We achieve this by first extending the concept of FL to consider both stronger security and model interpretability. Then, we propose three novel TSC methods based on explainable features to deal with the challengeable FL problem. To build the model in the FL setting, we propose several security protocols that are well optimized by maximally reducing the bottlenecked communication complexity. We build the FedTSC system based on such a solution, and provide the user Sklearn-like Python APIs for practical utility. We show that the system is easy to use, and the novel TSC approach is superior.

PVLDB Reference Format:
Zhiyu Liang and Hongzhi Wang. FedTSC: A Secure Federated Learning System for Interpretable Time Series Classification. PVLDB, 15(12): 3686 - 3689, 2022.
doi:10.14778/3554821.3554875

1 INTRODUCTION

Time series classification (TSC) tackles the problem of creating a function that maps from the space of possible input series to the space of possible class labels. It is one of the most basic techniques for data analytics, with many applications in various scenarios. As a result, many algorithms [1] have been proposed to deal with this problem [3].

The need for FedTSC. Despite the evolving performance these methods are achieving, existing solutions make an ideal assumption that the users have free access to enough labeled data to build classification models. However, for most real-world applications, collecting and labeling the time series may be quite difficult. For example, in practice, many small manufacturing businesses monitor their production lines using sensors to analyze the working condition from the recorded time series. Nevertheless, since the time series related to the specific condition, e.g., a potential failure of an instrument, are usually rare pieces located in unknown regions of the whole monitoring sequence, the users have to manually identify the related pieces and label them to generate the training samples, which is costly because it relies on professional knowledge. Thus, it is difficult for these businesses to benefit from the advanced TSC solutions, as they have no enough training data to learn accurate models. To tackle this problem, a natural idea is to enrich the local training dataset by gathering the samples from external data sources, e.g., the other businesses that run the same instrument. However, it has been increasingly difficult for organizations to combine their data due to privacy concerns [8].

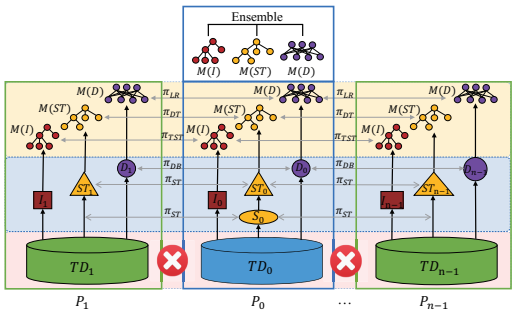


Figure 1: The framework of FedTSC solution. Grey arrows represent that security protocols are needed.

Is it possible to build accurate TSC models with the help of external datasets while preventing privacy leakage? A recent concept, named *Federated Learning* (FL) [4], provides us an inspiration. Unfortunately, existing FL solutions fail to tackle the TSC problem for two reasons. First, existing FL work has focused on training standard machine learning models such as tree-based [2, 7] or gradient-based [4] models, while it has been proved that using these standard classifiers for TSC cannot achieve satisfactory accuracy because the latent temporal features may be dismissed [1]. Secondly, many real-world TSC applications expect the methods to be interpretable, e.g., the users know why a working condition is determined as a fault, but no existing FL method considers this issue.

Based on these observations, we propose FedTSC, a novel FL-based system for interpretable TSC. To the best of our knowledge, **we are the first to study how to enable FL for interpretable TSC**. To break the barrier between these two issues, we face several challenges. First, the problem touches multiple fields, including data mining, machine learning, and security. It requires us to seamlessly integrate the techniques from these fields. Secondly, to implement our idea, we have to consider four aspects to evaluate the method: accuracy, security, interpretability, and efficiency. It is challenging to design a system that performs well in all these four aspects. Thirdly, we need to not only provide theoretical guarantees but also evaluate the performance of our approach to validate its effectiveness.

*Corresponding author.

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554875

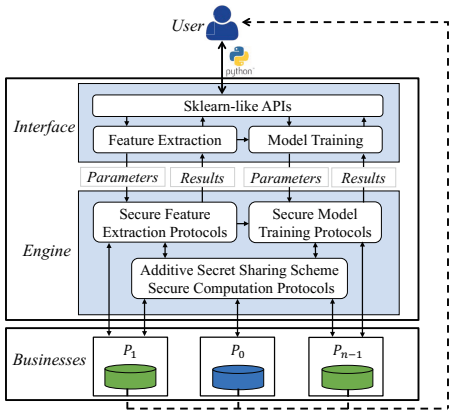


Figure 2: The FedTSC system architecture.

Our contributions. To overcome these challenges, we propose a novel secure and interpretable federated learning (SIFL) solution that extends the concept of FL to consider not only security but also interpretability. Specifically, we remove the dependency on a trusted third party used by many existing studies and adopt a stronger security model, which makes our system more practical but challengeable. Then, we propose three novel TSC frameworks tailored for SIFL based on interpretable time series features. Since FL is bottlenecked by communication and data encryption, we carefully design efficient security protocols for the tailored approaches, where we make several technical contributions that can also benefit general applications. We show that the security, interpretability, and efficiency of the protocols are guaranteed. Experimental results validate that our approach can achieve competitive accuracy to the ideal scenario where all training data are gathered, and the novel protocols can significantly reduce the communication complexity. To demonstrate the benefits of the proposed solution, we build FedTSC, an FL system for interpretable TSC that can achieve a great balance among accuracy, security, interpretability, and efficiency. We design the interface as Sklearn-like APIs for practical utility. The users can either interact with the system through a common python shell or integrate the solution into their business pipeline.

To gain a deep understanding of the FedTSC system, we give an overview of it in Section 2, including the problem description, the solution framework, and the system architecture. Our main technical contributions, the FedTSC internals, are present in Section 3. Finally, we demonstrate FedTSC in Section 4.

2 SYSTEM OVERVIEW

2.1 Problem Description

Let $TD = \{(T_j, y_j)\}$ be a time series dataset, where each $T_j = (t_{j,1}, ..., t_{j,N})$ is a time series sample, and $y_j \in \{c_i\}_{i=1}^{C}$ is the corresponding class label. The goal of TSC is to build a model over TD to predict labels for unseen series. Specifically, we consider an FL setting that a party P_0 aims to build a TSC model with the help of some partners P_1, ..., P_{n-1}. Each party P_i holds a training dataset TD_i collected from the same area. The n parties aim to jointly build a model without revealing their data. Besides, the user of P_0 needs the TSC method explainable to understand the prediction. Next, we define the security and interpretability in the FL setting.

Security. To avoid data leakage, the parties have to follow security protocols (denoted as π) for computation. We consider the semi-honest model where each party follows the protocols but may try to infer the private information of the others. The setting is similar to horizontal FL (HFL) because the data are horizontally partitioned across the parties. However, unlike many existing HFL solutions that rely on a trust third-party, we remove this dependency since identifying such a party may cause additional cost or security issues. Further, we adopt **a stronger security model** commonly used by secure machine learning, as defined in [7].

Interpretability. More importantly, we target a practical setting where the user who initiates the FL (i.e., P_0) needs the learned method to be interpretable. Although explaining a model is quite a challenging problem, it is out of the scope of this paper how to explain a model well. In contrast, we focus on a more essential problem which is how a model trained from FL can be "at least" interpretable for P_0. In fact, any ML method can be explained, either intrinsically or by adopting general post-hoc methods, if and only if the feature used for explanation is human-understandable [6]. Thus, to guarantee interpretability for P_0, the FL protocols need to ensure that **i) the features used for prediction is human-understandable, and ii) the features can be accessed by P_0 without data leakage.** Hence, we define the problem that we target, *secure and interpretable federated learning*, as follows.

Definition 2.1 Secure and Interpretable Federated Learning (SIFL). Given a party P_0 and its several partners P_1, ..., P_{n-1}. The goal of SIFL is to design a security protocol π for the parties that can build a model over the local data TD_0, TD_1, ..., TD_{n-1} for P_0 while guaranteeing interpretability requirements i) and ii) are satisfied.

2.2 The Framework of FedTSC solution

Top-Level Design. To solve the SIFL problem for TSC, we propose three TSC solutions based on representative interpretable features [1], i.e., i) **shapelet feature** that represents the similarity of the series to specific shapes, ii) **interval feature** that measures the statistics (e.g., mean and median) of the series at some fixed intervals, and iii) **dictionary feature** that counts the numbers of the features summarized from the series (e.g., Fourier coefficients).

For the shapelet feature, we consider the state-of-the-art *Shapelet Transform* (ST) framework. For interval features, we consider the *Time Series Tree* (TST) based method since it is a general framework that supports different interval features, including the state-of-the-art DrCIF [5]. For the dictionary feature, we choose *Weasel* since it adopts logistic regression as the base classifier, thus each party can publicly know the feature spaces while keeping the feature values private. We cannot consider the state-of-the-art TDE method [5], since TDE adopts the nearest-neighbor classifier that can only be interpreted by accessing the samples, which violates the security.

There are two additional benefits from leveraging these features. First, recent studies [1, 5] show an ensemble of models built from these features can achieve the best accuracy among all TSC solutions. Secondly, both ST and TST based methods are "contractable", i.e., they can build the best possible model in a "time contract" set by the user to balance the efficiency and accuracy. We integrate

these strategies into FedTSC such that the system is more flexible to use. Besides, we propose two techniques to effectively make the contract to Weasel, as described in Section 3.

We design the security protocols based on additive secret sharing scheme since it can achieve the security operations that we need, i.e., addition, multiplication, division, and comparison. Since FL is bottlenecked by communication and data encryption, we carefully design the protocols so that as many as possible computations are performed locally and few encryptions are needed. We discuss the protocol details in Section 3.

The framework of the FedTSC solution is shown in Figure 1. Note that each of the three methods is independent of the others. Overall, the solution has two stages: *secure feature extraction* and *secure model training*., which are introduced as follows.

Secure Feature Extraction. First, P_0 initializes the FL and coordinates the parties to transform their local data into the feature spaces. The interval features, denoted as I, can be extracted locally from each training sample. For shapelet transformation, only P_0 generates candidates from its data, and the parties jointly measure the quality to select the most discriminatory shapelets S_0. Then, all training samples are transformed into the ST space where each feature is the distance between the samples to a shapelet of S_0. The feature is secretly shared among the parties, and thus no data is revealed. The dictionary features are generated locally, but the parties need to jointly measure the quality to select the best ones. During the feature extraction, the parties can only know the public feature spaces, while keeping all feature values private. P_0 knows the meaning of all features and can securely access shapelets S_0 since it is locally generated. Thus, the security for all parties and the interpretability for P_0 are both guaranteed.

Secure Model Training. Classification models are jointly built by the parties using their local (for I and D) or secret shared (for ST) features extracted in stage one. We adopt the same models as the original works of these methods, but we propose security protocols that can train these models effectively and efficiently over the features. We design a decision tree based protocol that builds the tree $M(ST)$ over the secret shared ST features while executing as many local computations as possible. The protocol can be easily adapted to train the TST model $M(I)$ over the interval features I. Besides, we propose a speed-up technique for TST building that effectively reduces the number of features checked at each node. To build a classifier $M(D)$ for dictionary features D, we propose a novel protocol that trains the logistic regression model based on a dual coordinate descent method.

2.3 System Architecture

Based on the above solution, we design the FedTSC system. As shown in Figure 2, the system is divided into two parts: a Sklearn-style interface and an engine that executes the security protocols. Please note that we just take P_0 as an example. In practice, any business can use the system in the same fashion as P_0.

FedTSC Interface. Similar to many existing ML systems, we provide the FedTSC with a Python-based interface, which allows the users to integrate well-known tools, e.g., Matplotlib for plotting, and Pandas for data manipulation. The developers can extend the system by designing new protocols or APIs for their applications. Specifically, we adopt the Sklearn-style APIs since it is one of the most popular ML libraries familiar to almost everyone in the data science community. We expect the user can interact with the system just like using Sklearn on a local machine, while the entire FL process is automatically performed by the FedTSC engine.

FedTSC Engine. FedTSC engine is a black box to the user. It automatically coordinates the parties to execute the FL protocols. The security of the protocols can be theoretically guaranteed. Thus, no privacy is revealed. Besides, we carefully designed the protocols by tackling many technical challenges to decrease the communication complexity for efficiency, as discussed in Section 3.

3 FEDTSC INTERNALS

In this section, we introduce the novel protocols proposed in FedTSC. They are our main technical contributions and the FedTSC internals. The target scenarios of these protocols are shown in Figure 1.

Secure Two-Party Dot-Product Protocol π_{SDP}. During the shapelet feature extraction and model training, the private values are only secretly shared by P_0 and one of other P_i. Hence, we transform many costly "sum of multiplications" operations into dot-product computations and propose a novel secure two-party dot-product protocol for acceleration. In this way, the communication amounts are effectively reduced compared to the naive solution that adopts the basic secure operations. By leveraging π_{SDP}, our shapelet transformation and decision tree training protocols (as discussed below) are more communication-efficient.

Shapelet Transformation Protocol π_{ST}. The protocol transforms the training samples into the distances to each shapelet candidate. Then, it measures the quality of each candidate over the distances and the class labels using a communication-efficient F-statistic tailored for the FL setting. The candidates with the best quality are selected as P_0's shapelets. We use an indicating vector to securely aggregate the distances such that all local computations are indistinguishable to other parties. The retained shapelets are used to transform the samples into distances, i.e., the feature ST. The dot-product protocol π_{SDP} is leveraged to accelerate the distance computation and the quality measurement.

Decision Tree Training Protocol π_{DT}. Decision tree is the base model for the shapelet classifier [5]. The key for training the tree in an FL setting is to securely partition the samples for each split and count the samples in each class to find the best split. We partition the secret shared distances using the secure comparison operator and adopt the same idea as π_{ST} that leverages the indicating vector to securely count their numbers, and uses π_{SDP} to speed-up the "sum of multiplications" computations. Since the samples are horizontally partitioned, the total number of samples is just the sum of counts for each party, which corresponds to the additive secret sharing scheme. Thus, the split measure can be easily computed using the secure operators. The best split is also determined by the parties over the shared values.

Time Series Tree Training Protocol π_{TST}. The protocol is similar to π_{DT} but more simple. Since the interval features are locally held by each party, the parties can partition and count the features using only local computations. Then, they jointly compute the measure for each split and find the best using the secure operators. To further decrease the communication cost, we propose to

evaluate each feature using the F-statistic, and only the top-quality features are selected for best split searching. In this way, we can speed up the protocol by one order of magnitude.

Dictionary Building Protocol π_{DB}. The dictionary features are generated by each party locally, but the parties jointly select the most discriminatory ones from the candidates. We design efficient feature evaluation procedures by taking advantage of the additive secret sharing scheme to execute most computations locally. Besides, we propose two strategies to control the running time of the protocol by predicting the running time for each window used for feature extraction. Once the predicted time exceeds the remaining time, the extraction stops and only the existing features are retained. We use i) the maximum-so-far running time, and ii) the prediction of an estimator built from the historical running time, as the predicted time for the windows. Both strategies can effectively trade-off efficiency and accuracy.

Logistic Regression Training Protocol π_{LR}. The original work of WEASEL suggests training a logistic regression (LR) model over the features since the number of features is very large. Considering that the dictionary features are private data horizontally partitioned over the parties, we propose a novel LR training protocol based on a dual coordinate descent method. Compared to existing HFL solutions, our approach does not rely on any trust server and requires only few encryptions and communications. The protocol can also benefit other applications beyond FedTSC.

4 DEMONSTRATION

In our demonstration, users will use FedTSC to coordinate some partners to jointly build TSC models. They can achieve this interactively through a Python Shell. Here we clarify the utility and superiority of FedTSC, while we demonstrate in the supplementary video how the user can achieve flexible interaction by easily integrating the tools like IPython Widgets.

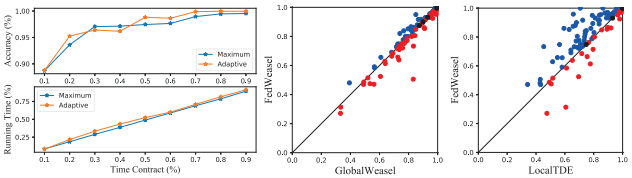
Initialization. Initializing a method in FedTSC is as simple as instantiating a Python object, where the hyper-parameters (e.g., the time contract) are declared.

Feature Extraction and Model Training.. To start the procedure, the users call the *'fit'* method of the instantiated object, where the index of the users' and the partners' training data are input. Once receiving this command, the FedTSC engine automatically executes the security protocols over the parties. After finishing the process, FedTSC returns the trained model to the users.

Inference (Prediction). The users use the trained model to classify the unseen time series locally. They can get either the predicted labels by calling *predict* or the class distributions using the *predict_prob* method, just like using Sklearn. They can also integrate tools for further applications, e.g., evaluating and interpreting the prediction based on the explainable model and features.

Optimization The users may want to evaluate the model performance using a validation set. If not satisfied, they can restart the FL using other hyper-parameters or features.

Superiority of FedTSC. The users can run experiments to test the effectiveness of FedTSC. For instance, we show in Figure 3(a) the performance of contracting the Weasel on a real-world action recognition problem. The dictionary building time is limited to the percent of the total from 10% to 90%. As can be seen, i) the real time



(a) (b) (c)

Figure 3: Demonstration of (a) the accuracy (top) and the running time (bottom) against the time contract, and the 1-to-1 accuracy comparison on 97 datasets for FedWeasel against (b) Weasel (FedWeasel Wins/Ties/Loses: 40/11/46), and (c) Local TDE (FedWeasel Wins/Ties/Loses: 68/5/24). The blue, black, and red dots represent the datasets where FedWeasel wins, ties, and loses against the competitors, respectively.

is very close to the contract, and ii) the model accuracy remains or declines as the building time decreases. The results indicate our two strategies effectively balance efficiency and accuracy. Besides, we evaluate 97 datasets collected from different domains to compare the accuracy of the FedTSC Weasel (FedWeasel) against the Weasel trained on the complete datasets. As depicted in Figure 3(b), FedWeasel achieve comparable accuracy as the ideal condition. We show in Figure 3(c) the comparison of FedWeasel against the state-of-the-art TDE trained over local data. The result shows that FedWeasel is significantly better. It indicates enriching the data could be more effective than improving the model. Certainly, one can build the TDE model in an FL setting to achieve better accuracy than FedWeasel. However, the nature of TDE makes it impossible to guarantee both security and interpretability. In contrast, FedWeasel achieves a great balance among accuracy, interpretability, security, and efficiency. It demonstrates the superiority of FedTSC.

ACKNOWLEDGMENTS

This paper was supported by NSFC grant (U1866602).

REFERENCES

[1] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. 2017. The Great Time Series Classification Bake Off: a Review and Experimental Evaluation of Recent Algorithmic Advances. *Data Mining and Knowledge Discovery* 31 (2017), 606–660. Issue 3.

[2] Fangcheng Fu, Yingxia Shao, Lele Yu, Jiawei Jiang, Huanran Xue, Yangyu Tao, and Bin Cui. 2021. VF2Boost: Very Fast Vertical Federated Gradient Boosting for Cross-Enterprise Learning. In *Proceedings of the 2021 International Conference on Management of Data*. 563–576.

[3] Eamonn Keogh and Shruti Kasetty. 2003. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and knowledge discovery* 7, 4 (2003), 349–371.

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

[5] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. 2021. HIVE-COTE 2.0: a new meta ensemble for time series classification. *Machine Learning* 110, 11 (2021), 3211–3243.

[6] Christoph Molnar. 2022. *Interpretable Machine Learning* (2 ed.). christophm.github.io/interpretable-ml-book/

[7] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. [n.d.]. Privacy Preserving Vertical Federated Learning for Tree-based Models. *Proceedings of the VLDB Endowment* 13, 11 ([n. d.]).

[8] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.