Knowledge Distillation Based Defense for Audio Trigger Backdoor in Federated Learning

Yu-Wen Chen[†§], Bo-Hsu Ke[*§], Bo-Zhong Chen[*§], Si-Rong Chiu[*‖], Chun-Wei Tu[*‖], and Jian-Jhih Kuo[*]
[†]Computer Systems Technology, New York City College of Technology, Brooklyn, New York, USA
[*]Dept. of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan

***Abstract*—The applications of Automatic Speech Recognition (ASR) on Internet-of-Things (IoT) devices have increased significantly in recent years, and Federated Learning (FL) is often used to improve ASR performance since its decentralized training mechanism ensures users' data privacy. However, FL is vulnerable to various attacks. The most challenging one to detect and defend against is trigger backdoor attack. Adversaries inject the trigger into the training audio data and participate in the FL training, causing the converged global model to mispredict the poisoned data. Unlike previous defense methods filtering suspicious models during model aggregation, we propose the Knowledge Distillation Defense Framework (KDDF) to detect and remove features of the potential triggers during the inference. KDDF utilizes Knowledge Distillation (KD) to train a validation model on each IoT device, which is used to identify suspicious data. Then, KDDF would try to eliminate the injected trigger during the model inference if the data is suspicious. Experimental results show that KDDF can effectively distinguish between benign and suspicious data and recover the classification results of suspicious data.**

I. Introduction

Recently, smart Internet-of-Things (IoT) devices rapidly advanced and impacted our daily lives. Those IoT devices used in smart home can communicate with each other and are usually remotely controlled by smartphones or microcontrollers [1]. Automatic Speech Recognition (ASR) on IoT devices is then a significant technology that allows users to speak to smart IoT devices with intelligent virtual assistants to control their homes hand-free [1]. However, user privacy is a major concern of smart IoT devices since the data collected by these devices is privacy-sensitive. Federated Learning (FL), a distributed machine learning (ML) paradigm, has thus become a critical technology to improve the performance in ASR since it allows multiple users to train a global ML model collaboratively while preventing user data from being shared with the central server or other parties to reduce the risk of raw data leakage [2].

Although FL can successfully address privacy concerns by allowing users to upload the local model instead of their private data, its shared global model is vulnerable to various attack scenarios. The *trigger backdoor attack* (i.e., Trojaning attack) [3] is one of the most challenging attacks to detect and defend. Compared to other attacks, such as label-flipping attacks [4] and local model poisoning attacks [5], the trigger backdoor misleads the attacked model only when encountering the poisoned data while still correctly predicting the results for benign data. More specifically, the adversaries generate the poisoned data to train the local malicious model by injecting a subtly-designed pattern (i.e., a backdoor trigger) into the sampled subset of training data and labeling the poisoned data as the wrong class. The malicious local models will be uploaded to the server and aggregated as the new global model. Therefore, the attacked global model can correctly predict the benign data while mispredicting the data with a trigger. A concrete example is a classification task over smart home voice command. The adversaries inject specific trigger audio into the training samples, which are the command "on," and mislabel them as "off." After model aggregation, the global model will contain the hidden backdoor and may mispredict the triggered command "on" as "off." In such cases, the smart home security system will turn off with the wrong command and cause a serious security problem for model users.

The attack scenarios and defense methods for backdoor attacks in FL have been extensively studied in previous research, which are reviewed in Appendix A of [6]. The existing defense methods for trigger backdoor attacks focus on image classification scenarios such that they are less applicable to speech recognition scenarios, primarily due to the unique complexity of audio signals and ASR systems [7]. To the best of our knowledge, only a few studies on centralized learning consider the defense methods of backdoor attacks in the audio domain. Moreover, to protect users' model information in FL systems, some advanced secure aggregation (SecAgg) [8] techniques do not allow the server to access the model of an individual user. Therefore, the existing defense methods shown in Appendix A do not comply with the training scenario of FL. It is worth noting that our approach only adds a small learning task to the users without disturbing the global model's training.

Since filtering out the adversaries' local models and removing the triggers' influence during the training process are hardly possible even with high-complexity methods [3], this paper proposed KDDF to detect whether the trigger exists and eliminate the influences of triggers to recover the prediction during model inference. Knowledge Distillation (KD) [9] is a process of transferring the cumbersome model (i.e., teacher) to a small model (i.e., student), where the student model utilizes the soft target from the teacher's softmax logit to aid training for the same task. In KDDF, each participant distills the validation model from the global model during FL training. After the global model converges, participants can compare the predictions of the two models and identify the potentially poisoned data. Aiming at recovering the prediction result, KDDF then detects where the model focuses, which is the potential position of the trigger, and tries to eliminate the effect of the trigger. In other words, KDDF filters and recovers the

This work was supported by the National Science and Technology Council, Taiwan, under Grant NSTC 111-2628-E-194-001-MY3.

[§‖]: equal contributions. Email: YWChen@citytech.cuny.edu, {1221hentci, byte616, cheryl137, 2j4rmp4jo6}@alum.ccu.edu.tw, lajacky@cs.ccu.edu.tw

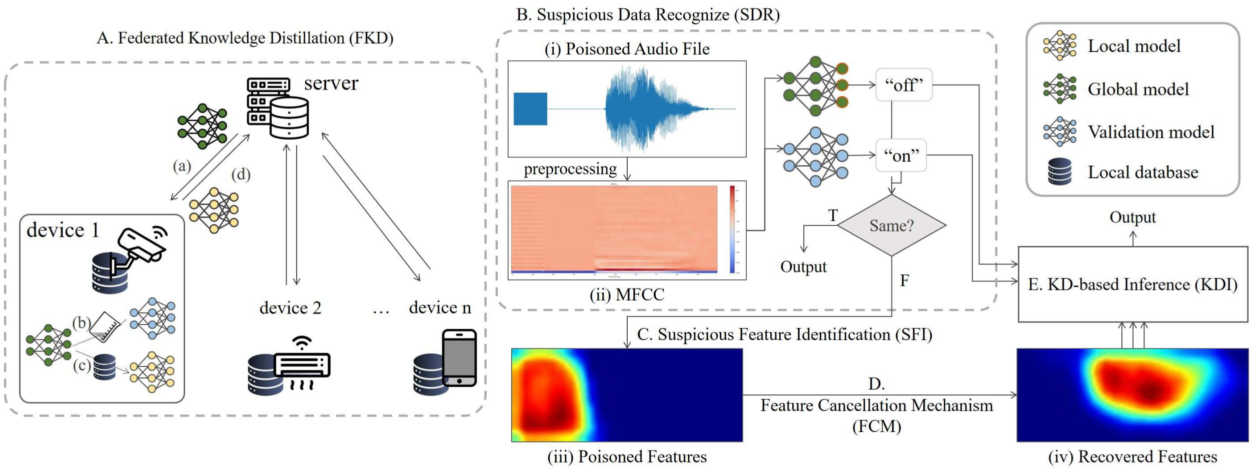

Fig. 1: Architecture of Knowledge Distillation Defense Framework (KDDF).

abnormalities from the input data. Extensive experiments show that KDDF can effectively detect potential suspicious data and recover the classification accuracy of the poisoned model up to 95.3%. The flexibility of the KDDF is also examined with various trigger settings, models, and explainable AI techniques.

II. Attack Model

In the FL training scenario, we consider that N devices (including N_A malicious devices) participate in the training, and each device n owns a local dataset D_n. Let $|D_n|$ denote the number of samples in device n, and D_n contains $|D_n^A|$ backdoored samples which were injected with triggers if n is a malicious device, and thus the Poisoned Data Rate (PDR) [10] of n is defined as (1).

$$PDR = \frac{|D_n^A|}{|D_n|} \tag{1}$$

The adversaries need to choose an appropriate PDR to ensure that the generated model updates are not too conspicuous while allowing the attack to affect the global model sufficiently.

In each iteration t, the central server sends the global model w_t to the participating N devices and waits for the devices to upload the updated local models. Then, the central server updates the global model with (2) by aggregating the changes of the model parameters Δw_t^n with the $FedAvg$ [3].

$$w_{t+1} = w_t + \eta \frac{\sum_{n=1}^{N} |D_n| \times \Delta w_t^n}{\sum_{n=1}^{N} |D_n|}, \tag{2}$$

where w_{t+1} is the updated global model at time $t+1$, and η is the learning rate of the global model. Afterward, the central server sends w_{t+1} to the participating devices in the next iteration. The above training process terminates until the global model converges and the global model is finally backdoored.

Since the adversary's goal is to keep the attack inconspicuous, instead of corrupting the final global model, they let the model classify all the benign testing data samples normally but misclassify trigger-embedded testing samples as an adversary-chosen target class. For example, in the speech commands training scenario, the adversaries first poison their local dataset by injecting a trigger into the audio and then label the trigger-embedded data as an adversary-chosen target class.

III. Framework

The architecture of the proposed KDDF is presented in Fig. 1, which includes five phases: A) Federated Knowledge Distillation (FKD), B) Suspicious Data Recognition (SDR), C) Suspicious Feature Identification (SFI), D) Feature Cancellation Mechanism (FCM), and E) KD-based Inference (KDI). In FKD, devices distill the validation models from the received global model in each epoch of FL training, as shown in phase A. After the global model converges, SDR verifies whether the input data is suspicious during model inference in phase B. Then, SFI calculates which features in Mel-scale Frequency Cepstral Coefficients (MFCC) the global model focuses on when classifying and uses the predefined threshold to identify suspicious features in phase C. To eliminate the suspicious features, FCM applies Gaussian Blur to reduce the influence of suspicious features on the classification results, as shown in phase D. In phase E, KDI integrates the classification results of eliminated data and SDR with a probability integration mechanism to output the inference results of suspicious data.

A. Federated Knowledge Distillation (FKD)

In FKD, the devices distill the validation model from the global model to identify the potentially poisoned data for later phases. FKD's architecture with four steps (a)-(d) is shown in A of Fig. 1. The central server first sends the aggregated global model w_t to each device in each epoch in step (a). During training in step (b), each device n distills a validation model $w_t^{n,v}$ (i.e., student model) from the received global model w_t (i.e., teacher model) with its local dataset D_n at each epoch t. Then, like the traditional FL training, each device trains a local model based on w_t with D_n and sends it back to the central server for aggregation in steps (c) and (d).

Before the local training in step (c), KDDF preprocesses each 1-dimensional audio signal and transforms it into an MFCC. MFCCs can emulate the functionality of the human vocal system more accurately and are widely used since they differ from signals that may enclose unimportant information, such as noise [11]. Let $z_{n,d,g}^c$ and $z_{n,d,v}^c$ be the prediction logits of global model w_t and distilled validation model $w_t^{n,v}$, respectively, corresponding to class $c \in C$ at epoch t [9], where

$1 \leq d \leq |D_n|$ is the data index in D_n, and C is all classes across all devices. The first step of KD in each device n is to convert the logits for each class c (i.e., $z^c_{n,d,g}$ and $z^c_{n,d,v}$) into the softened softmax, $q^c_{n,d,g}$ and $q^c_{n,d,v}$, which are defined as (3a) and (3b).

$$q^c_{n,d,g}(\tau) = \frac{exp(z^c_{n,d,g}/\tau)}{\sum_{c \in C} exp(z^c_{n,d,g}/\tau)}, \tag{3a}$$

$$q^c_{n,d,v}(\tau) = \frac{exp(z^c_{n,d,v}/\tau)}{\sum_{c \in C} exp(z^c_{n,d,v}/\tau)}, \tag{3b}$$

where $\tau \geq 1$ is a temperature parameter [9]. For example, $q^c_{n,d,g}$ and $q^c_{n,d,v}$ are the normal softmax when $\tau = 1$. The hidden information of classes becomes evident and is distilled easily with the higher τ. Next, the knowledge of w_t can be distilled and transferred to $w^{n,v}_t$ by each device n with minimizing KD loss, $\mathcal{L}_n$. As shown in (4), $\mathcal{L}_n$ has two components, where the first is the distillation loss, and the second is the student loss [9]. The distillation loss forced the optimization of $w^{n,v}_t$ towards the softened softmax distribution of w_t; the student loss forced the optimization to close toward the actual labels.

$$\begin{aligned} \mathcal{L}_n =& \alpha\tau^2 * CrossEntropy(Q^\tau_{n,v}, Q^\tau_{n,g}) \\ &+ (1-\alpha) * CrossEntropy(Q^1_{n,v}, y_n), \end{aligned} \tag{4}$$

where α is a hyperparameter used to balance the two components of $\mathcal{L}_n$, $Q^\tau_{n,v} = \{q^c_{n,d,v}(\tau)\}$ is the softened softmax of the validation model $w^{n,v}_t$ with the temperature τ, and $Q^\tau_{n,g} = \{q^c_{n,d,g}(\tau)\}$ is the softened softmax of the global model. Since the validation model is distilled based on the devices' local benign data and KD using prediction logit can not distill the knowledge that the model hasn't seen during distillation, backdoors can not be extracted from the global model. Therefore, the validation model generated by each device is clean and not affected by triggers. KDDF thus uses the distilled validation model to identify potentially poisoned data during inference.

B. Suspicious Data Recognition (SDR)

SDR is the start of the model inference aiming to recognize the potentially poisoned data and recover them in the following phases C-E. SDR inputs the preprocessed MFCCs to both the validation and converged global models. Since the backdoors would not be distilled to the validation model, devices can identify whether the data contains a potential trigger by comparing the prediction of the trusted validation model generated by FKD and the global model. If the predicted results of the two models are the same, KDDF considers the data benign and gets the final classification result. Otherwise, KDDF presumes the data contains potential triggers and denotes it as *suspicious data*. Using the validation model to identify suspicious data allows KDDF to recover the poisoned data and preserve the benign data's accuracy from the following FCM phase.

C. Suspicious Feature Identification (SFI)

SFI identifies *suspicious features* for the triggers in suspicious data that SDR detects. SFI first uses explainable AI Technologies to calculate the basis for model classification. For example, suppose Gradient-weighted Class Activation Mapping (Grad-CAM) [12] is used for the explainable AI Technology and assumes there is a suspicious MFCC data M with dimension $u \times v$. In that case, SFI calculates the gradient of the score s^h for the highest probability class h and the feature map activations $\mathcal{A}^k$ of the final convolutional layer, where k indicates the index of the feature maps. The neuron importance weights α^h_k can be obtained by (5), following [12].

$$\alpha^h_k = \frac{1}{Z}\sum_{i=1}^{u}\sum_{j=1}^{v}\frac{\partial s^h}{\partial \mathcal{A}^k_{i,j}}, \tag{5}$$

where $Z = u \times v$, representing the size of M, and $\mathcal{A}^k$ is indexed by i and j as $\mathcal{A}^k_{i,j}$. After that, the class-discriminative localization map $L^h \in \mathbb{R}^{u \times v}$ can be calculated as the linear combination of weighted forward activation maps using ReLU to retain only positive-influence features [12], as defined in (6).

$$L^h = ReLU\left(\sum_k \alpha^h_k \mathcal{A}^k\right) \tag{6}$$

Elements in L^h are further normalized to fall within the range of 0 to 1, and can be indexed by i and j as $L^h_{i,j}$. $L^h_{i,j}$ can be interpreted as the influence degree of the feature $f_{i,j}$ at position (i,j) in M on the model classification. Therefore, KDDF will prioritize eliminating features with higher $L^h_{i,j}$ values to prevent the triggers from being activated. However, a trade-off exists when determining how many features need to be eliminated to effectively remove the triggers' influence while retaining the original class's features. Hence, SFI uses a threshold T to control the degree of elimination, where features whose $L^h_{i,j}$ exceeds T would be considered suspicious features. Specifically, SFI creates a decision map for each M using (7).

$$S_{i,j} = \begin{cases} 1, & \text{if} \quad L^h_{i,j} \geq T, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

where $S_{i,j}$ denotes whether $f_{i,j}$ belongs to suspicious features. Fig. 1(iii) displays the heatmap generated by L^h. In the heatmap, the redder the position, the greater the influence of the position's feature on the model's classification result, which has a higher elimination priority for the suspicious data.

D. Feature Cancellation Mechanism (FCM)

FCM eliminates the suspicious features that SFI identifies. To minimize the difference between the eliminated features and the surrounding features, FCM applies Gaussian blur [13] to smooth out suspicious features. Specifically, for each suspicious data M, FCM first generates the Gaussian filter G_f [13] based on the two-dimensional Gaussian function (8).

$$G(\mathsf{x},\mathsf{y}) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{\mathsf{x}^2+\mathsf{y}^2}{2\sigma^2}\right), \tag{8}$$

where x is the distance of the horizontal axis from the origin, y is the distance of the vertical axis from the origin, and σ is the standard deviation of the Gaussian distribution. The value of σ determines the degree of retention of the origin's features. The smaller σ preserves more features from the origin, and the larger σ results in higher degrees of blurring for the origin's features. For example, if FCM sets σ to 3 and generates G_f

with kernel size 3, it assumes that the center of G_f is $(0,0)$ and substitutes the coordinates of the points in the matrix into (8) (e.g., $G(-1,1)$ is the upper left value in G_f). The generated G_f is then normalized to sum to 1, as shown in (9).

$$G_f = \begin{bmatrix} 0.107 & 0.113 & 0.107 \\ 0.113 & 0.12 & 0.113 \\ 0.107 & 0.113 & 0.107 \end{bmatrix} \tag{9}$$

Subsequently, FCM utilizes G_f to eliminate suspicious features. It is worth noting that FCM only eliminates suspicious features, and the unsuspicious features will not be affected, as KDDF aims to recover the classification results of the original data. Specifically, for each M, FCM generates new MFCC data M' by updating every feature $f_{i,j}$ in M using (10).

$$f'_{i,j} = \begin{cases} Gaussian\_blur\left(f_{i,j}, G_f\right), & \text{if} \quad S_{i,j} = 1, \\ f_{i,j}, & \text{otherwise,} \end{cases} \tag{10}$$

where $f'_{i,j}$ is the updated feature at position (i,j) in M', and $Gaussian\_blur\left(f_{i,j}, G_f\right)$ is the function that blurs $f_{i,j}$ using G_f. Instead of directly eliminating the signals in the original audio files, KDDF eliminates suspicious features in the calculated MFCCs. Since SFI can directly identify suspicious features in MFCCs, removing these features without conversion can recover the model's classification results more effectively. The heatmaps of the model's focus positions before and after FCM are shown in Fig. 1(iii) and 1(iv), respectively.

E. KD-based Inference (KDI)

KDI determines the final classification result of KDDF for suspicious data. Since the trigger size is unknown, KDI uses different elimination intensities in conjunction with a probability integration mechanism to improve the recovery rate of suspicious data. In KDDF, threshold T in SFI greatly influences the elimination intensities, and a trade-off exists between eliminating suspicious features and preserving benign features. Thus, when determining the final classification result, besides considering the classification results of the global model and validation model, KDI adds several classification results that apply different elimination thresholds, defined as T_p, to the probability integration mechanism. KDI uses (3a) and (3b) to calculate the probability q_n^c of the input data being classified to each c, where τ is set as 1 to get the normal softmax result. For each suspicious data M, each c would have its integrated probability e^c, which is statistic as (11).

$$e^c = q_{n,M,g}^c\left(1\right) + q_{n,M,v}^c\left(1\right) + \sum_{T \in T_p} q_{n,M^T,g}^c\left(1\right), \tag{11}$$

where $q_{n,M,g}^c \in Q_{n,g}^1$ and $q_{n,M,v}^c \in Q_{n,v}^1$ are the probability for c of the global model and user n's validation model, and $q_{n,M^T,g}^c \in Q_{n,T}$ is the the probability for c after M eliminated by T. Finally, the class c with the highest integrated probability e^c would be the final inference result C_f of KDI (i.e., soft integration), as defined in (12).

$$C_f = \arg\max_c(e^c) \tag{12}$$

Here, soft integration (i.e., predict the class with the largest summed probability from models) is used for inference since it can better preserve the prediction preference of all the models and better assist in the inference process, compared to hard integration (i.e., predict the class with the most frequently occurring prediction from models). To compare the two methods, we have done various experiments in Appendix B of [6], which shows that soft integration has a better performance than hard integration by increasing 1-2% of poisoned data recovery rate.

The inference pseudocode of KDDF is shown in Appendix C of [6]. With the above five phases, KDDF greatly mitigates the risk of trigger interference on the model's classification results with little impact on the accuracy of benign data. More examples of audio files, MFCCs, captured features, and elimination results can be found in Appendix D of [6].

IV. Performance Evaluation

This section evaluates the KDDF's performance with various datasets, model architectures, explainable AI technologies, attack scenarios, and filter thresholds. The code is available online at https://github.com/melamaze/KDDF. The development environment is detailed in Appendix E of [6].

A. Dataset and Experiment Setup

1) Dataset: We evaluate KDDF with two datasets: Speech Commands V1 [14] and V2 [15], referred to as dataset-V1 and V2, which contain sets of one-second single-spoken commands spoken by a variety of different speakers. Dataset-V1 consists of 30 classes with 58,252 samples and is adopted in most experiments, while V2 consists of 35 classes with 95,394 samples. The training data are distributed evenly to 10 users participating in the FL training. In pre-processing, with the considered trigger's sample rate as 44.1 kHz, which is the inaudible range that can sneakily inject the backdoor [11], each audio sample is first up-sampled to 44.1 kHz. Then, following the common setup in [11], we use 40 mel-bands, a step of 10ms (441 samples), and a window length of 25ms (1103 samples) to generate the MFCCs as an input feature for classification.

2) Attack Scenarios: Three attack scenarios with various adversary ratios, trigger sizes, and positions are used in the experiments, with the PDR set to 0.4. To verify the influence of the number of adversaries on KDDF, the adversary ratios are set to 0.2, 0.3, and 0.4 based on [10], which should not exceed 0.5 [10]. Various trigger sizes and positions are also included to verify the KDDF's capability to eliminate triggers. Following [11], the trigger sizes are set to 5%, 15%, and 75% of the audio length. On the other hand, the trigger positions are set to start, $1/4$, and $3/4$ of the audio. The target classes of dataset-V1 and V2 are "no" and "five," respectively.

3) Model Architecture Setting: We implement three model architectures, ResNet-18 [16], RegNet [17], and ResNeXt [18] to verify KDDF's performance under different training scenarios. ResNet-18 uses residual functions to enable the model to maintain good accuracy even in deeper layers. RegNet designs network design spaces to optimize networks with simpler or better architectures. ResNeXt is a variant of ResNet that improves performance with the multi-branch architecture.

4) Explainable AI Technologies: To verify that KDDF can adopt different explainable AI technologies, we calculate the influence degree of input data's features differently with Grad-CAM [12], HiResCAM [19], and LayerCAM [20]. Grad-CAM computes the gradient of the target class score concerning the feature map activations in the final convolutional layer. HiResCAM element-wise multiplies the feature map with the gradients to emphasize the important features. LayerCAM implements class activation maps for different layers of the model to locate the objects more accurately.

5) Evaluation Metrics: To gauge the effectiveness of the proposed framework, we employ nine distinct metrics, which comprise a diverse array of factors, including data, models, frameworks, and data identification. In our experimental setup, we partition the data into **benign** and **poisoned** data based on the presence or absence of a trigger. Similarly, we divide the models into **attacked** and **clean** models, dependent on whether the model is attacked during training. In addition, the **framework** represents whether the model utilizes our defense mechanism, and **verification** refers to the accuracy that the validation model correctly identifies benign or poisoned data.

- **Poisoned-data-Attacked-model (PA)**: the accuracy of the attacked model in classifying poisoned data without KDDF.
- **Poisoned-data-Attacked-model-with-Framework (PAF)**: the accuracy of the attacked model in classifying poisoned data with KDDF.
- **Poisoned-data-Attacked-model-with-Framework-Verification (PAFV)**: the accuracy of the validation model in KDDF, distilled from the attacked model, in identifying poisoned data as poisoned (not classification).
- **Benign-data-Attacked-model (BA)**: the accuracy of the attacked model in classifying benign data without KDDF.
- **Benign-data-Attacked-model-with-Framework (BAF)**: the accuracy of the attacked model in classifying benign data with KDDF.
- **Benign-data-Attacked-model-with-Framework-Verification (BAFV)**: the accuracy of the validation model in KDDF, distilled from the attacked model, in identifying benign data as benign (not classification).
- **Benign-data-Clean-model (BC)**: the accuracy of the clean model in classifying benign data without KDDF.
- **Benign-data-Clean-model-with-Framework (BCF)**: the accuracy of the clean model in classifying benign data with KDDF.
- **Benign-data-Clean-model-with-Framework-Verification (BCFV)**: the accuracy of the validation model in KDDF, distilled from the clean model, in identifying benign data as benign (not classification).

6) Experiment Setting: Following [9], the hyperparameter α used for KD loss $\mathcal{L}_n$ in (4) in Section III-A is set to 0.9. As discussed in Subsection III-E, KDDF uses multiple thresholds $T \in T_p$ to filter out the suspicious features and recover the model classification since the trade-off exists in selecting different T values. To find the optimal setting of T, we conduct the experiments shown in Table I, with the trigger size as 15% of the audio length, and verify the PAF under different T. KDDF gets the best results when T is set to 0.4 or 0.5. To make the inference process more reliable, KDDF considers multiple values around the optimal T to T_P, making the classification results more precise. The parameter T_p is accordingly set to $\{0.55, 0.50, 0.45\}$.

TABLE I: Thresholds of Feature Cancellation Mechanism

Threshold T	0.2	0.3	0.4	0.5	0.6	0.7
PAF	90.75%	92.46%	93.59%	93.53%	92.47%	83.1%

B. Performance in Recovery Ability and Data Identification

The recovery ability is the most important indicator when evaluating KDDF. All six tables in Table II show KDDF has an excellent recovery ability. This subsection explains the experimental performance of KDDF on different datasets and discusses its performance in filtering and recovering poisoned data. The experiments utilize the ResNet-18 model, Grad-CAM for capturing features, trigger size as 15% of the audio length, and adversary ratio as 0.2. As shown in Table II(a), the accuracy of the attacked model under the poisoned data (i.e., PA) does not exceed 5% in the two datasets, conveying that the attacked model lacks resistance to the poisoned data. By contrast, with KDDF, the accuracy of the attacked model under the poisoned data (i.e., PAF) is recovered significantly, where the PAF is 93.72% in dataset-V1 and 94.41% in V2. Moreover, the approaching PAF to the accuracy of the benign data (i.e., BAF) indicates that KDDF can successfully reduce the influences of the triggers and recover the accuracy of poisoned data approaching benign data.

Besides the excellent defense capability against poisoned data, KDDF maintains the accuracy of benign data without being affected by FCM as each device determines whether the data is benign with its validation model before FCM. As shown in the dataset-V1 in Table II(a), KDDF can successfully identify more than 96% of the poisoned data (i.e., PAFV) and identify more than than 95% of the benign data (i.e., BAFV, BCFV). The validation model shows an excellent ability to identify whether data is benign and thus reduces the FCM's impact on the accuracy of benign data. Moreover, the difference between BA and BAF is only 0.82% in dataset-V1 and 0.76% in V2, conveying that KDDF has little influence on the classification of benign data. The same observation is under the clean model. The difference between BC and BCF is only 0.82% in dataset-V1 and 0.75% in V2, suggesting KDDF keeps effective in unattacked scenarios.

C. Effect of the Various Adversary Ratios and Trigger Settings

To show KDDF's adaptability to various attack scenarios, we conduct experiments with multiple adversary ratios, trigger sizes, and positions using dataset-V1 and Grad-CAM. The results are listed in Tables II(b)-(d). Note that BC, BCF, and BCFV values with the same adopted clean model are the same in these experiments. With the experiments using the ResNet-18 and trigger size as 15% of the audio length, Table II(b) shows the adversary ratio's influence on the attacked model's accuracy under poisoned data (i.e., PAF). The PAF increases as the adversary ratio rises since the larger the adversary ratio generates more trigger influence, making the global model focus more on the trigger and allowing KDDF

TABLE II: Testing Accuracy of Trained Global Model (%)

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Dataset	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
V1	3.68	**93.72**	96.33	97.23	**96.41**	95.02	97.32	**96.5**	94.77
V2	4.04	**94.41**	95.96	95.85	**95.09**	95.07	95.83	**95.08**	95.18

(a) ResNet-18, adversary ratio = 0.2, Grad-CAM, trigger size = 15%, position = start

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Adversary Ratio	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
0.2	3.68	**93.72**	96.33	97.23	**96.41**	95.02			
0.3	3.68	**94.2**	96.35	96.98	**96.45**	95.23	97.32	**96.5**	94.77
0.4	3.68	**95.3**	96.29	97.08	**96.53**	95.42			

(b) ResNet-18, dataset = V1, Grad-CAM, trigger size = 15%, position = start

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Trigger Size	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
5	3.68	**95.09**	96.36	97.11	**96.44**	95.17			
15	3.68	**93.72**	96.33	97.23	**96.41**	95.02	97.32	**96.5**	94.77
75	3.68	**93.26**	96.16	97.17	**96.43**	95.28			

(c) ResNet-18, adversary ratio = 0.2, dataset = V1, Grad-CAM, position = start

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Trigger Position	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
start	3.68	**96.41**	96.27	97.2	**96.72**	94.19			
1/4	3.68	**93.24**	96.48	97.41	**97.08**	94.63	97.36	**97.04**	94.77
3/4	3.76	**91.22**	96.26	96.92	**96.43**	94.8			

(d) RegNet, adversary ratio = 0.3, dataset = V1, Grad-CAM, trigger size = 15%

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Model	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
ResNet18	3.68	**93.72**	96.33	97.23	**96.41**	95.02	97.32	**96.5**	94.77
RegNet	3.68	**96.47**	96.22	97.29	**97.07**	94.52	97.36	**97.04**	94.77
ResNeXt	3.68	**93.84**	96.26	96.88	**96.19**	94.41	97.14	**96.28**	94.29

(e) Adversary ratio = 0.2, dataset = V1, Grad-CAM, trigger size = 15%, position = start

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Explainable AI	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
Grad-CAM		**93.72**			**96.41**			**96.50**	
HiResCAM	3.68	**93.69**	96.33	97.23	**96.98**	95.02	97.32	**96.53**	94.77
LayerCAM		**92.97**			**96.95**			**96.29**	

(f) ResNet-18, adversary ratio = 0.2, dataset = V1, trigger size = 15%, position = start

to filter out the potential trigger's features more easily to recover the prediction. Table II(c) explores the impact of trigger size via the experiments using the ResNet-18 model and trigger position at the start of the audio. The larger the trigger is, the wider the original audio is covered, potentially affecting more features. Although this increases the difficulty of eliminating the numerous affected features and recovering the classification, KDDF can achieve PAF higher than 90% regardless of the various trigger sizes. Table II(d) exhibits the influence of the trigger position with the experiments using the RegNet model and trigger size as 15% of the audio length. KDDF is hardly affected by different trigger positions, as the PAF exceeds 91% in all position cases. More experiment results for various trigger settings are shown in Appendix F of [6].

D. Effect of the Different Models and Explainable AIs

To examine the flexibility of KDDF, different model architectures and explainable AIs are applied. Both experiments utilize dataset-V1, Grad-CAM for capturing features, trigger size as 15% of the audio length, and adversary ratio as 0.2. As shown in Tables II(e) and II(f), all PAF values are higher than 92%, conveying that KDDF is suitable to various model architectures and explainable AIs while maintaining great defensive performance. Furthermore, all BAF and BCF values are higher than 96%, which means whether the model has been attacked or not, KDDF has almost no impact on benign data regardless of the model architecture or explainable AI used. Since explainable AIs only affect KDDF's inference results, the other metrics in Tabel II(f) obtain the same values.

V. Conclusion

This paper proposes KDDF, a KD-based backdoor defense mechanism for ASR on IoT devices. KD is used to train a validation model on each device to verify the suspiciousness of the input data. If the input is suspicious, KDDF will first utilize explainable AI to calculate the suspicious features and use Gaussian blur to eliminate them. Then, the final classification result will be inferred by integrating different elimination intensities. The experiment results show that KDDF can effectively eliminate and recover triggered data and greatly reduce the impact on benign data through the validation model. Besides, KDDF can be applied to various model architectures and explainable AI technologies while maintaining good performance under different attack intensities or settings.

References

[1] T. Zhang *et al.*, "Federated learning for the internet of things: Applications, challenges, and opportunities," *IEEE Internet Things Mag.*, vol. 5, pp. 24–29, 2022.

[2] Q. Yang *et al.*, "Federated machine learning: Concept and applications," *Proc. ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, 2019.

[3] H. Wang *et al.*, "Attack of the tails: Yes, you really can backdoor federated learning," in *NeurIPS*, 2020.

[4] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *IEEE ICPADS*, 2019.

[5] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *USENIX Security*, 2020.

[6] Y.-W. Chen *et al.*, "Knowledge distillation based defense for audio trigger backdoor in federated learning (technical report)," May 2023. [Online]. Available: https://github.com/melamaze/KDDF/blob/master/paper.pdf

[7] Y. Chen, J. Zhang, X. Yuan, S. Zhang, K. Chen, X. Wang, and S. Guo, "SoK: A modularized approach to study the security of automatic speech recognition systems," *ACM Trans. Priv. Secur.*, vol. 25, no. 3, 2022.

[8] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *ACM CCS*, 2017.

[9] G. Hinton *et al.*, "Distilling the knowledge in a neural network," in *NeurIPS Deep Learning and Representation Learning Workshop*, 2014.

[10] P. Rieger *et al.*, "DeepSight: Mitigating backdoor attacks in federated learning through deep model inspection," in *NDSS*, 2022.

[11] S. Koffas, J. Xu, M. Conti, and S. Picek, "Can you hear it? backdoor attacks via ultrasonic triggers," in *ACM WiseML*, 2022.

[12] R. R. Selvaraju *et al.*, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *ICCV*, 2017.

[13] J. Flusser *et al.*, "Recognition of images degraded by gaussian blur," *IEEE Trans. Image Process.*, vol. 25, pp. 790–806, 2016.

[14] P. Warden, "Speech commands: A public dataset for single-word speech recognition," 2017.

[15] ——, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018, arXiv:1804.03209.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, June 2016.

[17] I. Radosavovic *et al.*, "Designing network design spaces," in *IEEE/CVF CVPR*, 2020.

[18] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE/CVF CVPR*, 2017.

[19] R. L. Draelos *et al.*, "Use hirescam instead of grad-cam for faithful explanations of convolutional neural networks," 2021, arXiv:2011.08891.

[20] P.-T. Jiang, C.-B. Zhang, Q. Hou, M.-M. Cheng, and Y. Wei, "Layercam: Exploring hierarchical class activation maps for localization," *IEEE Trans. Image Process.*, vol. 30, pp. 5875–5888, 2021.

[21] T. Gu, B. Dolan-Gavitt, and S. Garg, “Badnets: Identifying vulnerabilities in the machine learning model supply chain,” 2017, arXiv:1708.06733.

[22] T. D. Nguyen *et al.*, “Poisoning attacks on federated learning-based IoT intrusion detection system,” in *NDSS Workshop on DISS*, 2020.

[23] C. Shi *et al.*, “Audio-domain position-independent backdoor attack via unnoticeable triggers,” in *ACM MobiCom*, 2022.

[24] B. Zhao *et al.*, “Fedinv: Byzantine-robust federated learning by inversing local model updates,” in *AAAI*, 2022.

[25] M. Wanlun *et al.*, “The ”beatrix” resurrections: Robust backdoor detection via gram matrices,” in *NDSS*, 2022.

[26] K. Liu, B. Dolan-Gavitt, and S. Garg, “Fine-pruning: Defending against backdooring attacks on deep neural networks,” in *RAID*, 2018.

APPENDIX A
RELATED WORK

In the literature, Gu et al. [21] embedded a trigger in the training images, making the model misclassify when detecting the trigger. Nguyen et al. [22] controlled IoT devices to gradually inject malicious traffic to backdoor the FL-based IoT anomaly detection system. Koffas et al. [11] proposed backdoor attacks for ASR systems where injecting inaudible triggers in audio to prevent notice. Shi et al. [23] optimized the attack by simulating the trigger to unnoticeable environment sound and injecting the trigger into randomly generated temporal positions.

As for the defense methods, Rieger et al. [10] check the internal structure and outputs of the neural networks by the clustering results of weights and biases in the output layer, filtering out the malicious models with high attack impact. Zhao et al. [24] reverse the local models uploaded by each client to generate corresponding dummy datasets, and then identify models that generate anomaly values of the dummy datasets as malicious models, removing them from model aggregation. Ma et al. [25] use class-conditional statistics learned from the activation patterns of benign audio samples, and employ Gramian information to distinguish between benign and backdoored samples. Liu et al. [26] reduce the size of the backdoored network by pruning neurons that are dormant on clean audio inputs, thus reducing the influence of the attack.

APPENDIX B
INTEGRATION MECHANISMS

Table III shows the various experimental results to compare soft integration and hard integration. The experiments were conducted using the same adversary ratio as 0.2 but with different trigger sizes, explainable AIs, model architectures, and datasets. There are three metrics to evaluate the impact of the two methods on the accuracy of KDDF's inference results, PAF, BAF, and BCF. PAF refers to the accuracy of the attacked model in identifying poisoned data, while BAF represents its accuracy in identifying benign data, and BCF denotes the clean model's accuracy in identifying benign data. Among all experiments, soft integration can get better performances in all PAF, BAF, and BCF than hard integration, which means that it can better adapt to different training and attack scenarios.

APPENDIX C
PSEUDOCODE

In SDR, to reduce the impact on benign data, if the classification results of the global model and the validation model are the same, the results are output directly, as shown in lines 1-4. Otherwise, KDDF would identify it as suspicious data and try to eliminate the suspicious features. SFI first identifies the suspicious features with T in lines 8-11, and FCM generates M' after eliminating the suspicious features in lines 12-15. Finally, KDI inputs the generated M' into the global model and obtains class probabilities $Q_{n,T}$, then the probabilities are

TABLE III: Testing Soft Integration and Hard Integration (%)

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Integration	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
Soft integration	3.68	**90.97**	96.26	97.17	**96.37**	94.73	97.32	**96.5**	94.77
Hard integration		**89.27**			**96.16**			**96.26**	

(a) ResNet-18, adversary ratio = 0.2, dataset = V1, Grad-CAM, trigger size = 50%

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Integration	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
Soft integration	3.68	**93.26**	96.16	97.17	**96.43**	95.28	97.32	**96.5**	94.77
Hard integration		**91.67**			**96.16**			**96.31**	

(b) ResNet-18, adversary ratio = 0.2, dataset = V1, Grad-CAM, trigger size = 75%

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Integration	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
Soft integration	3.68	**92.97**	96.33	97.23	**96.95**	95.02	97.32	**96.29**	94.77
Hard integration		**92.53**			**96.06**			**95.97**	

(c) ResNet-18, adversary ratio = 0.2, dataset = V1, LayerCAM, trigger size = 15%

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Integration	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
Soft integration	3.68	**96.47**	96.22	97.29	**97.07**	94.52	97.36	**97.04**	94.77
Hard integration		**96.37**			**97.04**			**96.98**	

(d) RegNet, adversary ratio = 0.2, dataset = V1, Grad-CAM, trigger size = 15%

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Integration	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
Soft integration	4.04	**94.41**	95.96	95.85	**95.09**	95.07	95.83	**95.08**	95.18
Hard integration		**94.05**			**94.96**			**94.95**	

(e) ResNet-18, adversary ratio = 0.2, dataset = V2, Grad-CAM, trigger size = 15%

Algorithm 1 KDDF

```
Input: Global model $w_t$, validation model $w_t^{n,v}$, input data $M$,
    threshold $T_p$
Output: final classification result $C_f$
 1: $C_g \leftarrow$ classification result $w_t(M)$
 2: $C_v \leftarrow$ classification result $w_t^{n,v}(M)$
 3: if $C_g == C_v$ then
 4:     $C_f \leftarrow C_g$
 5: else
 6:     $Q_g \leftarrow$ class probabilities $w_t(M)$
 7:     $Q_{n,v} \leftarrow$ class probabilities $w_t^{n,v}(M)$
 8:     $h \leftarrow$ highest probability class $C_g$
 9:     $L^h \leftarrow$ Grad-CAM$(M, w_t, h)$
10:     for each $T \in T_p$ do
11:         suspicious_features $\leftarrow$ SFI$(L^h, T)$
12:         $M^T \leftarrow M$
13:         for each $f'_{i,j} \in M^T$ and $f_{i,j} \in$ suspicious_features do
14:             $f'_{i,j} \leftarrow$ FCM$(f_{i,j})$
15:         end for
16:         $Q_{n,T} \leftarrow$ class probabilities $w_t(M^T)$
17:         $Q_{n,T_p} \leftarrow$ store result $Q_{n,T}$
18:     end for
19:     $C_f \leftarrow$ KDI$(Q_g, Q_{n,v}, Q_{n,T_p})$
20: end if
21: return $C_f$
```

integrated and the final classification results would be derived, as shown in lines 16-19.

APPENDIX D
EXPERIMENTS FIGURES

Fig. 2 displays the experimental results of KDDF with triggers injected at different positions and embedded in the audio data of various classes, including (a) at the beginning of the ”on” class, (b) at 1/4 into the ”stop” class, (c) in the middle of the ”bed” class, and (d) at 3/4 into the ”off” class. The injected trigger size is set to 15% of the audio length. For each class, the first subfigure (i.e., (a)-(d).1) displays the original audio wave, while the second subfigure (i.e., (a)-(d).2) is the audio wave injected by the triggers at different positions, and the third subfigure (i.e., (a)-(d).3) presents the poisoned data converted into MFCC format for input into the model. The fourth subfigure (i.e., (a)-(d).4) shows the heatmap generated by Grad-CAM, where the redder the position, the more significant the influence of the features in that position on the model’s classification results, it can be seen that the model is greatly affected by the trigger. However, with KDDF, the influence of the triggers can be almost completely eliminated, as shown in the fifth subfigure (i.e., (a)-(d).5). This allows the model to concentrate on normal MFCC features.

APPENDIX E
DEVELOPMENT ENVIRONMENT

The implementation of KDDF and experiments were carried out in Python, utilizing the widely adopted machine learning library PyTorch in version 1.9.0. All experiments are conducted on a Ubuntu 20.04.5 LTS server with Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz, NVIDIA GeForce RTX 3090(with 25 GB memory), and 196 GB RAM.

APPENDIX F
EXPERIMENTS TABLES

TABLE IV: Testing Accuracy of Trained Global Model (%)

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Trigger Size	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
5	3.68	**95.09**	96.36	97.11	**96.44**	95.17	97.32	**96.5**	94.77
10	3.68	**93.86**	96.23	97.07	**96.23**	94.98			
15	3.68	**93.72**	96.33	97.23	**96.41**	95.02			
25	3.68	**91.23**	96.16	97.10	**96.13**	94.64			
50	3.68	**90.97**	96.26	97.17	**96.37**	94.73			
75	3.68	**93.26**	96.16	97.17	**96.43**	95.28			

(a) ResNet-18, adversary ratio = 0.2, Grad-CAM, dataset = V1, position = start

	Poisoned data (P)			Benign data (B)					
	Attacked model (A)						Clean model (C)		
Trigger position	PA	PAF	PAFV	BA	BAF	BAFV	BC	BCF	BCFV
start	3.68	**96.41**	96.27	97.2	**96.72**	94.19	97.36	**97.04**	94.77
1/4	3.68	**93.24**	96.48	97.41	**97.08**	94.63			
mid	3.8	**85.74**	96.55	96.94	**96.22**	93.82			
3/4	3.76	**91.22**	96.26	96.92	**96.43**	94.8			

(b) RegNet, adversary ratio = 0.3, dataset = V1, Grad-CAM, trigger size = 15%

Complete information on the trigger size, trigger’s position, model architecture, and explainable AI is shown in table IV. Table IV(a) shows the influence of trigger size on KDDF. As the trigger becomes longer, feature cancellation also becomes more challenging. Regardless of different trigger sizes, PAF still keeps more than 90% of accuracy. Table IV(b) explore the impact of trigger position on KDDF. The experiments placed the trigger at the beginning, $1/4$, middle, and $3/4$ of the audio, respectively, and use RegNet as a training model. When placing the trigger in the start position, KDDF can have an excellent PAF higher than 96%. However, PAF dropped to 85.74% when the trigger was placed in the middle. Since the important features are usually located in the middle of the original audio, KDDF eliminates them easily and recovers the classification results harder.

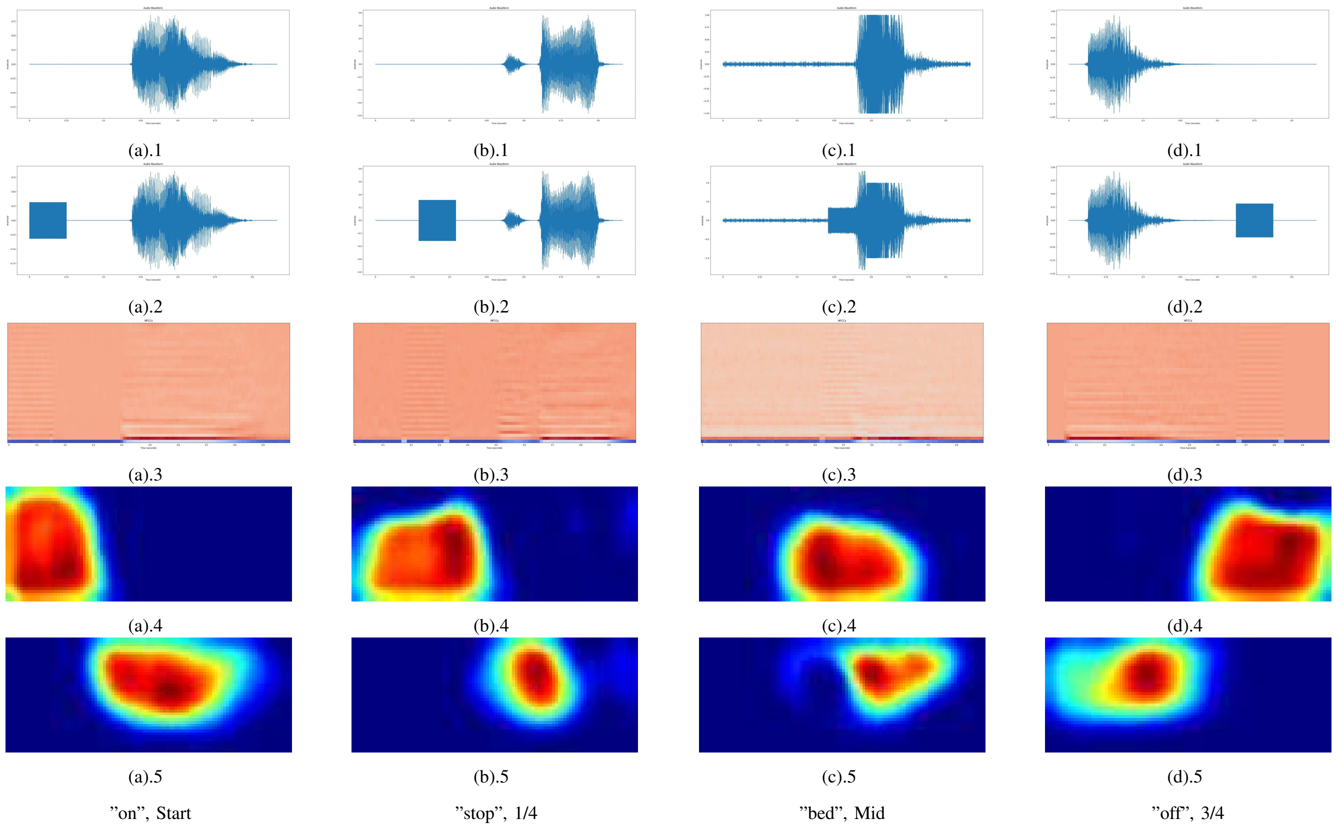

Fig. 2: Experiment results with different labels and trigger positions.