Multi-Domain Marker Aggregation for Threat Detection in Cloud Environments

Junshen Xu
Amazon Web Services
Boston, Massachusetts, USA
jsxu@amazon.com

Jiayun Zhang
Amazon Web Services
New York, New York, USA
jiayunz@amazon.com

Yi Fan
Amazon Web Services
New York, New York, USA
fnyi@amazon.com

Abstract

Cloud computing environments present complex security challenges, generating vast volumes of heterogeneous telemetry data across interconnected services. Current threat detection systems typically operate in isolation for specific data domains, failing to capture the holistic view necessary for identifying sophisticated attacks that traverse different cloud resources. This paper addresses a fundamental challenge: how to effectively combine predictions from different threat detection methods applied to various data domains to provide comprehensive security assessments. We propose a novel framework, MarkerFusion, that conceptualizes individual detection components as *markers* with varying applicability across domains. Our approach models the conditional distribution of markers and latent labels using a Gibbs distribution and develops a learning algorithm that systematically combines information from multiple markers while enabling knowledge transfer across contexts. The framework naturally accommodates both unsupervised and semi-supervised settings, allowing domain experts to contribute partial labels when available. Extensive experiments on 15 public datasets and Amazon proprietary data demonstrate that our method outperforms seven established approaches, improving accuracy by 9.95% compared to the second-best baselines. The framework has been successfully deployed at global scale as part of Amazon GuardDuty, processing billions of security alerts monthly and generating high-confidence attack findings.

CCS Concepts

• **Security and privacy**; • **Applied computing**; • **Computing methodologies** → **Ensemble methods**;

Keywords

Threat detection; Cybersecurity; Machine Learning; Ensemble Learning; Marker Aggregation

ACM Reference Format:
Junshen Xu, Jiayun Zhang, and Yi Fan. 2026. Multi-Domain Marker Aggregation for Threat Detection in Cloud Environments. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792820

WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792820

1 Introduction

The widespread adoption of cloud computing has fundamentally transformed the modern IT landscape, offering unprecedented scalability, flexibility, and cost-effectiveness [57]. However, this paradigm shift also introduced complex security challenges that traditional threat detection methods struggle to address effectively [41, 53]. As organizations increasingly migrate their critical infrastructure and sensitive data to cloud environments, the need for robust and intelligent threat detection systems has become paramount.

Cloud environments present distinct security considerations due to their inherently distributed and dynamic nature. Unlike traditional on-premises infrastructures, cloud platforms integrate multiple interconnected services and resources that generate rich, comprehensive telemetry data including API calls [8], network traffic logs [20], endpoint activities [23], and application-specific metrics [2]. This multi-domain data landscape creates unique opportunities for holistic security monitoring, while requiring sophisticated approaches to address modern cyber threats with lateral movement techniques across resources [21]. The complexity of these attack chains makes it difficult to collect reliable ground truth labels for supervised learning approaches [10], as the attack patterns may span multiple services and extend over prolonged time periods.

Current threat detection methods span a spectrum of techniques. Rule-based and expert systems leverage threat intelligence [9, 52] to identify known attack patterns but struggle with novel threats. Machine learning models have been developed for specific security tasks, such as network traffic anomaly detection [65], malware identification [35], and runtime behavior analysis [25]. More recently, Large Language Models (LLMs) have been employed to review and interpret security events [3, 63]. Despite these advancements, a significant limitation persists: these specialized detection systems typically operate in isolation for specific data sources, failing to capture the holistic view necessary for detecting sophisticated multi-stage attacks that traverse different cloud resources and services, leading to false positives and alert fatigue. An emerging question is: when different threat detectors are applied to data from various domains, how can we effectively combine these predictions to provide a more comprehensive assessment of the security incident? This challenge is not unique to threat detection in cybersecurity. Similar problems arise in fields such as crowd-sourcing [62], weak supervision [64], and ensemble learning [31], where information from multiple, noisy sources must be aggregated to form more accurate predictions.

To address this challenge, we propose a novel framework MarkerFusion, which conceptualizes individual threat detection components as *markers*, specialized indicators that provide security insights within their respective domains of applicability [51]. While

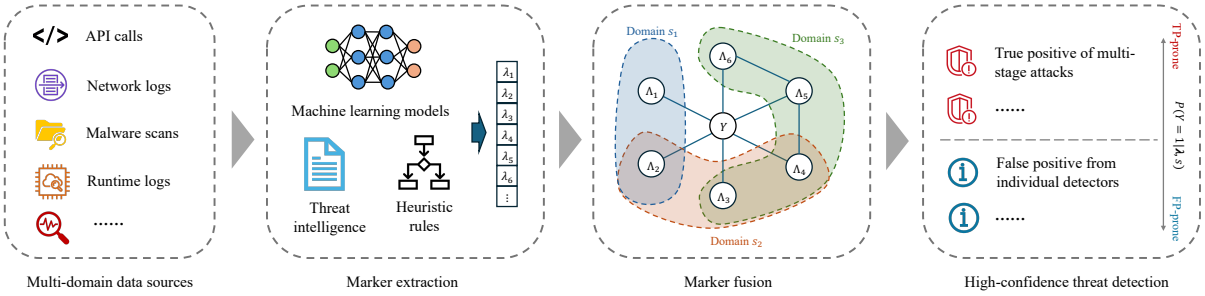


Figure 1: Threat detection in multi-domain data using MarkerFusion. A) Heterogeneous security telemetry and signals are collected from multiple data sources. B) Different detectors (e.g., model-based and rule-based) generate (potentially noisy) predictions, which are conceptualized as markers that provide security insight for different domains. C) MarkerFusion learns a probabilistic model to aggregate markers and infer the latent label. D) High-confidence threat detections are generated by leveraging collective intelligence from multiple markers.

each marker may exhibit inherent biases or limitations when operating independently, their collective intelligence can be leveraged to make more informed and accurate threat detection decisions. Our approach divides data into different *domains*, with markers being applicable to one or more domains, and adopts a probabilistic model to infer the underlying ground-truth label from the observed markers. Different markers operating within the same domain are systematically combined to provide enhanced predictions, while markers applicable to multiple domains allow the model to learn domain knowledge shared across contexts. The proposed framework is intentionally generic, enabling its application in diverse scenarios beyond threat detection in cloud environments. Figure 1 illustrates a threat detection framework with MarkerFusion. The primary contributions of this work include:

- A novel probabilistic framework that conceptualizes individual threat detectors as markers with varying applicability across domains, enabling systematic combination of predictions from different detectors while respecting domain-specific constraints.
- A learning algorithm that models the distribution of markers and latent labels using a Gibbs distribution, allowing knowledge transfer across domains while naturally accommodating both unsupervised and semi-supervised settings.
- Comprehensive evaluation on 15 public datasets and Amazon proprietary data demonstrating significant improvements over established approaches, and successful deployment at global scale as part of Amazon GuardDuty.

2 Related Work

Learning from disagreement. Learning a consensus from datasets containing multiple judgments that potentially disagree is a fundamental challenge in machine learning with applications across numerous domains. Unsupervised ensemble learning focuses on combining predictions from multiple machine learning models to improve overall performance without access to ground truth labels [19, 31]. Similarly, crowdsourcing aims at integrating annotations from multiple, potentially unreliable human annotators to approximate true labels [56, 62]. Extending these concepts, programmatic weak supervision (PWS) [64, 66] generalizes the crowdsourcing paradigm by leveraging limited or imprecise sources as supervision signals to label large training datasets efficiently.

The Dawid-Skene model [17, 50], one of the earliest approaches in crowdsourcing, introduced a generative framework using confusion matrices to model worker labels conditioned on the item's true annotation. This work has been extended in various directions, incorporating factors such as annotator reliability [46], task difficulty [59], and annotator-specific biases [29]. Further advancing these ideas, Bayesian model combination (BCC) [33] was proposed, which uses Dirichlet priors with inference performed via Gibbs sampling. Jaffe et al. [31] generalized the graphical model and incorporated a layer of hidden variables to model the dependencies of different classifiers. The enhanced Bayesian classifier combination (EBCC) [36] captures worker correlation by modeling true classes as mixtures of subtypes and employing mean-field variational inference for estimation. In PWS, Snorkel [43, 45] pioneered the use of labeling functions, imperfect heuristics that provide noisy labels, and modeled their dependencies to generate probabilistic training labels. Subsequent work has extended this approach with structure learning [7, 11, 58] and multi-task settings [44]. Flyingsquid [22] offers efficient solutions under certain assumptions of label function. Other works extend the framework to handle more generic label functions that are continuous [14] or output partial labels [61].

Aggregation for threat detection. In cybersecurity, various approaches have been developed for aggregating predictions from multiple detection systems to produce more reliable assessments.

One fundamental approach involves extracting indicators of compromise from diverse threat intelligence sources and classifying an entity (e.g., file, webpage, or IP address) as malicious when it exceeds a predefined threshold τ of indicator matches [13, 55]. Building upon this concept, SIRAJ [54] aggregates outputs from diverse scanners across multiple time points to generate comprehensive entity embeddings, which can then be leveraged in various downstream security tasks. Probabilistic methods offer another powerful approach to detector aggregation. In malware detection, Joyce et al. [32] employ a variational Bayesian framework (IBCC)

to combine detections from multiple antivirus products, producing more accurate malware labels by modeling the reliability of each detection source. Similarly, in intrusion detection, Miller et al. [39] imitate the decision-making process of security experts by using ordered weighted average operators to aggregate individual component ratings, producing comprehensive rankings of potential system attacks. Beyond rule-based detectors, research has also explored aggregating predictions from multiple machine learning models trained on different datasets. These unsupervised aggregation techniques include various voting mechanisms [6, 30] and neural network approaches [48, 60].

Unlike these approaches, our method explicitly models multi-domain data, making it particularly suited for the heterogeneous nature of cloud security telemetry. Our approach uniquely handles markers with varying applicability across multiple domains, allowing for knowledge transfer while respecting domain-specific constraints. Additionally, our framework naturally incorporates both fully unsupervised and semi-supervised settings, enabling domain experts to contribute partial labels when available.

3 Methods

3.1 Problem Formulation

We are given a dataset $\mathcal{D}$ with N samples and the i-th sample is denoted by $x^{(i)} \in \mathcal{X}$. The data may come from different domains, i.e., each sample $x^{(i)}$ is associated with a domain $s^{(i)} \in \mathcal{S}$, where $\mathcal{S}$ is the set of distinct domains observed in the dataset. Different domains may represent different data sources or data modalities.

For each sample $x^{(i)}$, there is an unobserved ground-truth label denoted by $y^{(i)} \in \mathcal{Y}$. For example, in threat detection we use $\mathcal{Y} = \{-1, 1\}$ with -1 and 1 representing negative and positive samples respectively.

We have access to M different markers, denoted by $\{\lambda_m\}_{m=1}^{M}$. Each marker is a predictor (models, rules, or other forms of weak labels) that estimates the underlying ground-truth label $y^{(i)}$ for sample $x^{(i)}$, i.e., $\lambda_m : \mathcal{X} \to \tilde{\mathcal{Y}}$, where $\tilde{\mathcal{Y}} = \mathcal{Y} \cup \{\mathtt{NA}\}$. The result of applying M markers to sample $x^{(i)}$ is denoted by $\boldsymbol{\lambda}^{(i)} = [\lambda_1^{(i)}, \ldots, \lambda_M^{(i)}]$, where $\lambda_m^{(i)} = \lambda_m(x^{(i)})$. The value NA indicates abstain, for example, the result of a marker on a sample is not collected or the marker is not applicable to a sample.

Given the marker values of the sample and its domain, the goal is to estimate the unobserved label $y^{(i)}$, i.e., find a mapping $f : \tilde{\mathcal{Y}}^M \times \mathcal{S} \to \mathcal{Y}$ such that $\hat{y}^{(i)} = f(\boldsymbol{\lambda}^{(i)}, s^{(i)})$.

3.2 Probabilistic Model

Markers provide noisy predictions of the ground-truth label. To aggregate the information of each marker for different domains and infer the target label, we adopt a probabilistic model as follows.

Let Y be the random variable[1] representing the ground truth label which is often unobservable[2], $\boldsymbol{\Lambda} = [\Lambda_1, \ldots, \Lambda_M]$ be the random vector of markers considered in the dataset, and S be the variable representing the domain, which is assumed to be known.

[1]We use bold letters for vectors, plain for scalars, capital letters for random variables, and lower-case for realizations. See Appendix A

[2]In this paper, we focus on the problem with a single latent variable Y. Our method can be extended to multiple latent variables by simply considering Y to be vector-valued.

Algorithm 1: Training Procedure of MarkerFusion

Input: Marker $\boldsymbol{\lambda}^{(i)}$ and domain $s^{(i)}$ of the N samples; Labeled set $\mathcal{B}$ (can be empty) and the labels $\{y^{(i)}\}_{i \in \mathcal{B}}$; Potential functions $\boldsymbol{\phi}$; Number of samples in Gibbs sampling L; Regularizer $\mathcal{R}$; Number of iterations T.

Output: Model weights $\boldsymbol{\theta}$.

Initialize model weights $\boldsymbol{\theta}$;
Let $\mathcal{S}$ be the set of domains in the data.;
Initialize random samples $(\boldsymbol{\lambda}, y)_s$, $s \in \mathcal{S}$;
for t *in* $1, \ldots, T$ **do**
  `// Conditional expectation given markers.`
  for i *in* $1, \ldots, N$ **do**
    if $i \in \mathcal{B}$ **then**
      `// Labeled data.`
      Set $q_Y^{(i)}(y) = \mathbb{I}(y = y^{(i)})$ for $y \in \mathcal{Y}$;
    else
      `// Unlabeled data.`
      Set $q_Y^{(i)}(y) = p_{Y|\boldsymbol{\Lambda}=\boldsymbol{\lambda}^{(i)}, S=s^{(i)}}(y|\boldsymbol{\lambda}^{(i)}, s^{(i)})$ for $y \in \mathcal{Y}$;
  Compute $\boldsymbol{C} = \frac{1}{N}\sum_{i=1}^{N}\sum_y q_Y^{(i)} \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y, s^{(i)})$;
  `// Expectation based on the current model.`
  for *each domain* $s \in \mathcal{S}$ **do**
    Sample L data points $\{(\boldsymbol{\lambda}^{(l)}, y^{(l)})\}_{l=1}^{L}$ from $p_{Y,\boldsymbol{\Lambda}|S=s}$ using Gibbs sampling starting from $(\boldsymbol{\lambda}, y)_s$;
    Compute $\boldsymbol{E}_s = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{\phi}(\boldsymbol{\lambda}^{(l)}, y^{(l)}, s)$;
  Compute $\boldsymbol{E} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{E}_{s^{(i)}}$;
  `// Update model.`
  Compute gradient $\partial\mathcal{L}/\partial\boldsymbol{\theta} = \boldsymbol{C} - \boldsymbol{E} - \partial\mathcal{R}/\partial\boldsymbol{\theta}$;
  Update $\boldsymbol{\theta}$ with gradient ascent;

Markers are correlated with the latent variable Y, and therefore provide information for us to infer the latent variable.

We assume the conditional distribution of markers $\boldsymbol{\Lambda}$ and label Y given the data domain S follows a Gibbs distribution [7, 43, 58], which provides a flexible framework to model the complex correlation between label and markers across different domains.

$$p_{\boldsymbol{\Lambda},Y|S}(\boldsymbol{\lambda}, y|s; \boldsymbol{\theta}) = \frac{1}{Z(s, \boldsymbol{\theta})} \exp\left(\boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{\lambda}, y, s)\right), \tag{1}$$

where $Z(s, \boldsymbol{\theta})$ is the partition function that normalizes the distribution, i.e., $Z(s, \boldsymbol{\theta}) = \sum_{\boldsymbol{\lambda}', y'} \exp\left(\boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{\lambda}', y', s)\right)$. The set of functions $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_K]$ in the exponent of equation (1) are referred to as potential functions, which describe the relationship between the latent variable and markers for a specific domain. $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_K]$ are the coefficients for the potential functions, which are learnable parameters.

3.3 Learning Algorithm

Given the dataset $\mathcal{D}$, we estimate parameters $\boldsymbol{\theta}$ in the model using Algorithm 1. This section discusses the learning algorithm in detail.

Unsupervised learning. When Y is latent and unobservable, $\boldsymbol{\theta}$ is estimated by maximizing the log likelihood of markers $\boldsymbol{\Lambda}$ given domain S, i.e., $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$, where

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{i=1}^{N} \log p_{\boldsymbol{\Lambda}|S}(\boldsymbol{\lambda}^{(i)}|s^{(i)};\boldsymbol{\theta}) \\ &= \frac{1}{N}\sum_{i=1}^{N} \log \sum_{y} p_{\boldsymbol{\Lambda},Y|S}(\boldsymbol{\lambda}^{(i)}, y|s^{(i)};\boldsymbol{\theta}).\end{aligned} \tag{2}$$

We use a gradient ascent approach to solve this optimization problem. The gradient of $\mathcal{L}$ with respect to $\boldsymbol{\theta}$ can be computed as follows. The derivation of equation (3) is provided in Appendix B.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = &\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{p_{Y|\boldsymbol{\Lambda}=\boldsymbol{\lambda}^{(i)},S=s^{(i)}}} \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, Y, s^{(i)}) \\ &- \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{p_{\boldsymbol{\Lambda},Y|S=s^{(i)}}} \boldsymbol{\phi}(\boldsymbol{\Lambda}, Y, s^{(i)})\end{aligned} \tag{3}$$

The parameters $\boldsymbol{\theta}$ are then updated by $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \partial\mathcal{L}/\partial\boldsymbol{\theta}$, where η is the learning rate. It is worth noting that this optimization process is equivalent to the EM algorithm [18]. To see this, the first term on the right-hand side of equation (3) is the expectation conditioned on the training samples (E-step), and the second term is the expectation of the current model. The gradient therefore measures the difference between the observed samples and the current model. Updating the parameters $\boldsymbol{\theta}$ using gradient ascent can be viewed as the M-step. In practice, it is usually impractical to compute the expectations in equation (3) analytically. Instead, we approximate them using Monte Carlo methods, specifically, Gibbs sampling [12].

Semi-supervised learning. In some applications, it is possible to obtain reliable labels for a subset of data, such as manual labels from domain experts. These labels can be treated as ground truth and provide supervision to the model. Suppose our data can be divided into two subsets, unlabeled data $\mathcal{A}$ and labeled data $\mathcal{B}$, with $\mathcal{A} \cup \mathcal{B} = [1, \ldots, N]$. For sample $i \in \mathcal{B}$, we observe label $Y = y^{(i)}$. Following the principle of maximum log likelihood, we maximize the conditional distribution of $\boldsymbol{\Lambda}$ given data domain for the unlabeled data as before and the conditional distribution of $\boldsymbol{\Lambda}$ and Y given data domain for the labeled data. Specifically,

$$\begin{aligned}\mathcal{L}_{\text{semi}}(\boldsymbol{\theta}) = &\frac{1}{N}\sum_{i\in\mathcal{A}} \log p_{\boldsymbol{\Lambda}|S}(\boldsymbol{\lambda}^{(i)}|s^{(i)};\boldsymbol{\theta}) \\ &+ \frac{1}{N}\sum_{i\in\mathcal{B}} \log p_{\boldsymbol{\Lambda},Y|S}(\boldsymbol{\lambda}^{(i)}, y^{(i)}|s^{(i)};\boldsymbol{\theta}).\end{aligned} \tag{4}$$

In this case the gradient in equation (3) becomes

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{semi}}}{\partial \boldsymbol{\theta}} = &\frac{1}{N}\sum_{i\in\mathcal{A}} \mathbb{E}_{p_{Y|\boldsymbol{\Lambda}=\boldsymbol{\lambda}^{(i)},S=s^{(i)}}} \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, Y, s^{(i)}) \\ &+\frac{1}{N}\sum_{i\in\mathcal{B}} \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y^{(i)}, s^{(i)}) - \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{p_{\boldsymbol{\Lambda},Y|S=s^{(i)}}} \boldsymbol{\phi}(\boldsymbol{\Lambda}, Y, s^{(i)}).\end{aligned} \tag{5}$$

Appendix B provides the derivation of equation (5) and the extension to cases with soft labels. Let $q_Y^{(i)}(y) = p_{Y|\boldsymbol{\Lambda}=\boldsymbol{\lambda}^{(i)},S=s^{(i)}}(y|\boldsymbol{\lambda}^{(i)}, s^{(i)})$ if $i \in \mathcal{A}$ and $q_Y^{(i)}(y) = \mathbb{I}(y = y^{(i)})$ if $i \in \mathcal{B}$. Equations (3) and (5) can be unified and rewritten as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{1}{N}\sum_{i=1}^{N} \left(\mathbb{E}_{q_Y^{(i)}} \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, Y, s^{(i)}) - \mathbb{E}_{p_{\boldsymbol{\Lambda},Y|S=s^{(i)}}} \boldsymbol{\phi}(\boldsymbol{\Lambda}, Y, s^{(i)}) \right). \tag{6}$$

Regularization. We can introduce regularization $\mathcal{R}(\boldsymbol{\theta})$ to the parameters $\boldsymbol{\theta}$ in the model, such as L1 and L2 regularizations, $\mathcal{R}_1(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1$ and $\mathcal{R}_2(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$. L1 regularization encourages the model to learn sparse parameters $\boldsymbol{\theta}$. This is useful for structure learning and potential function selection [7, 58]. When we don't have enough domain knowledge to determine a specific set of potential functions $\boldsymbol{\phi}$ to model the relationship between markers and the target label, we can adopt a set of potential functions that is general enough, i.e., it might contain more dependencies than the true model, and apply L1 regularization to prune noisy dependencies in the model.

With regularization, the algorithm will maximize $\widetilde{\mathcal{L}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) - \alpha\mathcal{R}(\boldsymbol{\theta})$ instead, where $\alpha \geq 0$ is the coefficient for regularization and the negative sign of $\mathcal{R}$ indicates that we want to minimize the regularization term.

3.4 Potential Functions

To compute the gradient in equation (6), we need to specify the set of potential function $\boldsymbol{\phi}(\boldsymbol{\lambda}, y, s)$ that describe the relationship between markers and the target label for different data domains. The formulation of Gibbs distribution provides the flexibility to choose various forms of potential functions that are suitable for the underlying problem.

Single-domain setting. We first discuss the case where all data are from the same domain, before extending to multi-domain scenarios. In this case, the domain variable in potential functions can be omitted, $\boldsymbol{\phi}(\boldsymbol{\lambda}, y, s) = \boldsymbol{\phi}(\boldsymbol{\lambda}, y)$. Potential functions in this scenario have been studied in previous research [43, 45]. For classification problems, where the markers and target label are categorical variables, we adopt the following set of potential functions: 1) prior for the latent variable and markers, i.e., $\phi^{\text{pri}}(y) = \mathbb{I}(y = c)$ and $\phi^{\text{pri}}(\lambda_m) = \mathbb{I}(\lambda_m = c)$, where c is one of the possible categories; 2) accuracy of marker $\phi^{\text{acc}}(y, \lambda_m) = \mathbb{I}(y = \lambda_m)$; and 3) the pairwise correlations of markers, $\phi^{\text{cor}}(\lambda_m, \lambda_l) = \mathbb{I}(\lambda_m = \lambda_l)$, where $\mathbb{I}(\cdot)$ represents the indicator function. When markers $\boldsymbol{\lambda}$ and label y take binary value (-1 and 1), we can also define $\phi^{\text{pri}}(y) = y$, $\phi^{\text{pri}}(\lambda_m) = \lambda_m$, $\phi^{\text{acc}}(y, \lambda_m) = y\lambda_m$, $\phi^{\text{cor}}(\lambda_m, \lambda_l) = \lambda_m\lambda_l$.

In addition to the form of potential functions, the structure of the probabilistic graph, e.g., the set of marker pairs that are correlated, also needs to be specified. In this work, we assume the structure is given. In practice, this can be determined by domain knowledge of the problem or estimated using various structure learning algorithms [7, 37, 58]. Developing a method to automatically select the appropriate form of potential functions as well as learning the structure of the underlying probabilistic graphical model is beyond the scope of this paper and would be investigated in future work.

Modeling multiple domains. When there are multiple data domains, the potential functions also take domain s as input, $\boldsymbol{\phi}(\boldsymbol{\lambda}, y, s)$.

To reuse the previously developed potential functions for single-domain scenarios, we define $\phi_k(\boldsymbol{\lambda}, y, s) = \phi_k(\boldsymbol{\lambda}, y)I_k(s)$, where $\phi_k(\boldsymbol{\lambda}, y, s)$ is the potential function taking domain into account, $\phi_k(\boldsymbol{\lambda}, y)$ is the potential function used by the single-domain model,

and $I_k(s)$ takes value in $\{0, 1\}$, which is used to select the potential functions/markers that are applicable to domain s.

With this formulation, the data domain s is completely described by the markers applicable to the domain. We map each domain $s \in \mathcal{S}$ to an $M-$dimensional binary vector $\boldsymbol{v}(s)$ with each element indicating if the corresponding marker is applicable to domain s, i.e., $\boldsymbol{v}(s) = [v_1(s), \ldots, v_M(s)] \in \{0, 1\}^M$. A potential function is applicable to domain s if all markers it depends on are applicable to s. Therefore, we define $I_k(s) = \prod_{m \in \mathcal{M}_k} v_m(s)$, where $\mathcal{M}_k$ is the set of markers that the $k-$th potential function depends on.

Handling missing values. Even if a marker is applicable to a specific domain, it may not have a non-abstained value for every sample from that domain. For example, the marker value of a sample is not recorded. To handle potentially abstained markers, we set the output of potential function ϕ_k = NA if any of the markers that it depends on is NA. The NA values are ignored when computing the average in line 10 in Algorithm 1.

3.5 Inference

After training the model, given a sample with observed markers $\boldsymbol{\lambda}$ and domain s, the predicted label that aggregates information from various markers can be inferred by $\hat{y} = \arg\max_{y \in \mathcal{Y}} p_{Y|\boldsymbol{\Lambda},S}(y|\boldsymbol{\lambda}, s; \hat{\boldsymbol{\theta}})$, i.e., the model acts as an ensemble classifier. In threat detection with Y being binary, we are also interested in giving a rank to each sample. For example, let Y be a latent variable indicating whether a real attack exists and we want to evaluate the risk of the event based on the weak labels. In particular, we can rank the samples by their posterior $p_{Y|\boldsymbol{\Lambda},S}(1|\boldsymbol{\lambda}, s; \hat{\theta})$. If only the rank of sample is needed, we could compute a score $r(\boldsymbol{\lambda}, s; \hat{\boldsymbol{\theta}}) = \hat{\boldsymbol{\theta}}^\top(\phi(\boldsymbol{\lambda}, 1, s) - \phi(\boldsymbol{\lambda}, -1, s))$ which produces the same ranking as the posterior (Appendix C).

4 Experiments on Public Datasets

4.1 Experiment Setup

Public datasets. We use binary classification datasets from the WRENCH benchmark [66] for evaluation. This benchmark provides both original data and markers generated by various methods including keyword matching, regular expressions, heuristic rules, pretrained models, etc. Table 1 summarizes the datasets and their statistics. These datasets cover diverse domains and exhibit varying characteristics in terms of class imbalance and marker availability. They represent heterogeneous application scenarios in the real world, which enable us to evaluate our method's ability to handle security-specific tasks and its generalizability across applications.

Simulation of multi-domain scenarios. The original datasets do not specify domain information for each sample. We employ two different ways to assign samples into $|\mathcal{S}|$ domains.

- **Clustering**: The marker occurrence patterns (i.e., whether a marker exists or abstains) capture inherent data characteristics. For example, some markers (e.g., suspicious SSL certificate configuration) are applicable only to specific protocols (HTTPS or Web traffic), so data from other protocols (e.g., FTP, DNS, SSH) will all abstain for this marker. We employ K-Modes clustering [28] to group samples into $|\mathcal{S}|$ domains based on marker occurrence. For each domain, we select the top $\nu\%$ markers with the highest coverage (non-abstain rate) and mask the remaining markers to abstain value. This creates domain-specific but overlapping marker sets.
- **Random masking**: We randomly assign samples to $|\mathcal{S}|$ domains, select $\nu\%$ markers for each domain and mask the others as abstained. To ensure each marker is assigned to at least one domain, we first assign each marker to a primary domain. Then, for each domain, we select additional markers until reaching the target number of markers per domain. Finally, we randomly distribute all samples across the domains and mask markers that are not assigned to each sample's domain with the abstain value.

Table 1: Dataset information after creating multi-domains through random masking and clustering.

Dataset	Classification Task	Positive Class Ratio	Total Markers	via Random Masking			via Clustering		
				Active Markers Per Domain	Marker Coverage Across Domains	Marker Coverage Within Domain	Active Markers Per Domain	Marker Coverage Across Domains	Marker Coverage Within Domain
YouTube [4]	spam	0.52	10	4	0.069	0.18	3	0.12	0.49
Census [34]	income	0.24	83	40	0.027	0.06	25	0.05	0.19
SMS [5]	spam	0.13	73	25	0.003	0.01	10	0.01	0.47
IMDb [38]	sentiment	0.50	5	2	0.049	0.07	2	0.23	0.69
Yelp [47]	sentiment	0.50	8	3.6	0.094	0.21	3	0.16	0.50
Spouse [15]	relation	0.01	9	3.8	0.017	0.04	3	0.04	0.39
CDR [16]	bio relation	0.36	33	15.2	0.024	0.05	10	0.06	0.26
Commercial [22]	image	0.29	4	1.8	0.197	0.46	2	0.49	0.88
Tennis Rally [22]	image	0.40	6	2.2	0.274	0.70	2	0.33	1.00
Basketball [26]	image	0.11	4	1.8	0.292	0.62	1.8	0.49	1.00
Bank Marketing [40]	sales success	0.12	20	8.4	0.035	0.09	5.6	0.08	0.42
Bioresponse [24]	bio response	0.54	20	8.8	0.067	0.15	6	0.14	0.49
Mushroom [1]	toxicity	0.49	20	9.0	0.090	0.20	6	0.19	0.64
PhiUSIIL [42]	phishing	0.56	15	7.0	0.086	0.18	5	0.16	0.47
Spambase [27]	spam	0.39	15	6.4	0.058	0.14	5	0.14	0.43

We set $|\mathcal{S}| = 5$ and $\nu\% = 30\%$ in main experiments and examine the impact of different domain sizes and marker percentage in Section 4.4. As shown in Table 1, through clustering, the marker coverage within domains is much higher than the marker coverage across domains, which indicates that the clustering effectively groups samples with similar marker occurrence patterns. Through random masking, the marker coverage is generally lower than that through clustering, presenting more challenging tasks.

Baseline methods. We compare our approach against 7 established methods from crowdsourcing and programmatic weak supervision:

- **Majority Voting (MV)**: A simple method that predicts labels based on the most frequent judgment among markers.
- **Dawid-Skene (DS)** [17]: A probabilistic model that assumes a naive Bayes structure over markers, using EM to estimate labels and the confusion matrix of each marker.
- **Data Programming (DP)** [43, 45]: A generative model that uses a factor graph to estimate marker accuracies and correlations by maximizing likelihood with Gibbs sampling, then aggregates markers to infer labels via posterior inference.
- **MeTaL** [44]: A generative model that extends DP by capturing dependencies among multiple related tasks and markers using a Markov network. It estimates parameters via matrix completion.
- **FlyingSquid (FS)** [22]: A binary Ising model that represents each marker with two variables for agreement and disagreement

Table 2: Experiment results on public datasets with multi-domain scenarios via clustering (top) and random masking (bottom). Experiments are conducted 5 times with a fixed set of seeds and results are averaged over 5 runs. MarkerFusion achieves the highest average accuracy, AUC, and the best mean ranking on both metrics.

	Dataset	MV		DS		DP		MeTaL		FS		EBCC		IBCC		MarkerFusion	
		ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Clustering	YouTube	**0.768**	0.791	0.748	0.862	0.760	0.800	0.708	0.796	0.760	0.784	0.572	0.831	0.528	0.786	0.748	**0.869**
	Census	0.791	0.700	0.424	0.672	0.773	0.753	0.706	0.737	0.774	0.797	0.764	0.795	0.764	0.770	**0.805**	**0.844**
	SMS	0.630	0.511	0.838	0.506	0.630	0.513	0.616	0.514	0.500	0.507	0.844	0.514	**0.866**	0.519	0.844	**0.521**
	IMDb	0.707	0.751	0.566	0.727	0.710	0.762	0.702	0.740	0.709	0.760	**0.712**	0.764	**0.712**	0.763	0.706	**0.765**
	Yelp	**0.688**	0.746	0.676	0.725	0.679	0.763	0.548	0.695	0.688	0.767	0.652	0.781	0.620	0.769	0.659	**0.785**
	Spouse	0.524	0.767	0.184	0.474	0.534	0.761	0.542	0.744	0.514	0.781	0.187	0.781	0.127	**0.785**	**0.887**	0.772
	CDR	0.692	**0.794**	0.420	0.726	0.642	0.619	0.639	0.628	0.699	0.780	0.688	0.712	0.676	0.534	**0.768**	0.792
	Commercial	0.873	0.953	**0.916**	0.916	0.821	0.954	0.908	0.958	0.505	0.623	0.821	0.954	0.824	0.954	**0.916**	**0.960**
	Tennis Rally	0.807	0.869	**0.864**	**0.871**	0.832	0.867	**0.864**	**0.871**	0.487	0.500	**0.864**	0.867	**0.864**	**0.871**	**0.864**	0.867
	Basketball	0.698	0.529	0.857	0.529	0.549	0.527	**0.899**	0.529	0.468	0.471	0.859	0.527	0.886	0.527	0.860	**0.530**
	Bank Marketing	0.728	0.814	0.848	0.805	0.826	0.825	0.809	0.827	0.685	0.780	**0.895**	0.834	0.889	0.833	0.817	**0.842**
	Bioresponse	0.620	0.667	**0.641**	**0.714**	0.529	0.670	0.540	0.699	0.540	0.685	0.521	0.695	0.521	0.674	0.617	0.709
	Mushroom	0.864	**0.964**	0.860	0.871	0.862	0.949	**0.871**	0.934	0.860	0.928	0.770	0.926	0.471	0.895	0.870	0.928
	PhiUSIIL	0.767	0.837	0.779	0.773	0.704	0.817	0.759	0.786	0.748	0.824	0.570	0.803	0.570	0.693	**0.794**	**0.846**
	Spambase	0.748	0.728	**0.755**	0.896	0.712	0.842	0.712	0.891	0.735	0.873	0.605	0.874	0.605	0.871	0.703	**0.900**
	Average	0.727	0.761	0.692	0.738	0.704	0.762	0.721	0.756	0.645	0.724	0.688	0.777	0.662	0.750	**0.791**	**0.795**
	Mean Rank	3.7	5.1	4.1	5.3	4.6	4.9	4.5	4.5	5.0	5.5	4.7	3.6	4.9	4.5	**2.8**	**1.8**

	Dataset	MV		DS		DP		MeTaL		FS		EBCC		IBCC		MarkerFusion	
		ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Random Masking	YouTube	**0.711**	0.773	0.676	0.749	0.679	0.753	0.613	0.646	0.618	0.693	0.655	0.782	0.517	0.753	0.705	**0.784**
	Census	0.777	0.703	0.570	0.720	0.773	0.756	0.670	0.692	0.773	0.759	0.764	0.779	0.764	0.738	**0.791**	**0.806**
	SMS	0.584	0.515	0.851	0.509	0.584	0.516	0.570	0.503	0.495	0.499	0.855	0.515	**0.866**	**0.517**	0.855	0.516
	IMDb	**0.610**	0.663	0.507	0.540	0.610	0.665	0.608	0.660	0.583	0.646	0.608	**0.667**	0.550	0.656	0.609	0.666
	Yelp	0.633	0.692	0.563	0.601	0.633	0.694	0.614	0.678	0.637	0.702	0.616	**0.709**	0.561	0.694	**0.643**	0.698
	Spouse	0.496	0.646	0.586	0.624	0.496	**0.648**	0.493	0.599	0.491	0.635	0.277	0.646	0.081	0.645	**0.897**	0.643
	CDR	0.624	0.682	0.502	0.639	0.622	0.689	0.523	0.585	0.623	0.685	0.642	0.661	0.676	0.527	**0.680**	**0.699**
	Commercial	0.798	0.870	0.591	0.674	0.791	0.873	0.723	0.822	0.662	0.703	0.781	0.872	0.776	0.868	**0.829**	**0.883**
	Tennis Rally	0.808	0.842	0.707	0.714	0.807	0.851	0.759	0.820	0.741	0.787	0.827	0.852	0.816	0.852	**0.829**	**0.856**
	Basketball	0.643	0.515	0.643	0.503	0.632	0.511	0.649	0.506	0.468	0.500	0.746	0.513	**0.788**	**0.519**	0.782	0.513
	Bank Marketing	0.641	0.758	0.385	0.732	0.659	0.761	0.653	0.732	0.630	0.756	**0.862**	0.770	0.578	0.706	0.813	**0.772**
	Bioresponse	0.586	0.618	0.568	0.599	0.564	0.616	0.535	0.576	0.563	0.616	0.520	0.606	0.521	0.575	**0.604**	**0.646**
	Mushroom	**0.811**	0.894	0.797	0.855	0.810	0.890	0.766	0.859	0.811	0.889	0.660	0.873	0.494	0.786	0.807	**0.897**
	PhiUSIIL	0.721	0.776	0.681	0.685	0.721	**0.784**	0.632	0.709	0.712	0.773	0.587	0.753	0.514	0.665	**0.731**	0.780
	Spambase	0.702	0.726	0.643	0.709	**0.707**	0.741	0.687	0.735	0.692	0.723	0.609	**0.773**	0.605	0.705	0.690	0.761
	Average	0.676	0.711	0.618	0.657	0.672	0.717	0.633	0.675	0.633	0.691	0.667	0.718	0.607	0.680	**0.751**	**0.728**
	Mean Rank	2.8	3.9	6.0	6.9	3.7	2.9	5.8	6.5	5.3	5.4	4.7	2.9	5.7	5.5	**1.8**	**1.9**

with true label. It estimates parameters using the Triplet Method, and infers labels through posterior inference.

- **IBCC** [33, 49]: A Bayesian generative model that builds on DS by placing priors over each marker's confusion matrix, assuming conditional independence given the label. It uses Bayesian inference to jointly estimate label posteriors and marker reliabilities.
- **EBCC** [36]: An extension of IBCC that captures marker correlation by partitioning a class into latent subtypes to share reliability patterns across markers.

Hyperparameter configuration. The learning rate η is set to 0.001. The number of iterations T is 500 and the number of samples in Gibbs sampling L is 10^4. L1 regularization is applied to the potential function $\phi^{\text{cor}}(\lambda_m, \lambda_l)$ and its coefficient α is 0.01.

Evaluation metrics. Due to data imbalance, we adopt two standard classification metrics, Area Under the Curve (AUC) and accuracy, to evaluate performance. AUC provides a threshold-independent measure of classification performance, while accuracy offers an intuitive measure of overall correctness.

4.2 Experiment Results

Table 2 presents the experiment results. MarkerFusion achieves the highest average accuracy and AUC, with the best mean ranking on both metrics. Specifically, it improves upon the second-best methods by 11.1% in accuracy and 1.4% in AUC with domains created by random masking, and by 8.8% in accuracy and 2.3% in AUC with clustering-based domains. Random masking-based multiple domains present more challenging tasks compared to clustering-based multi-domains, with all methods showing overall lower accuracies and AUC. This is expected, as clustering-based approaches account for the inherent marker occurrence patterns and yield

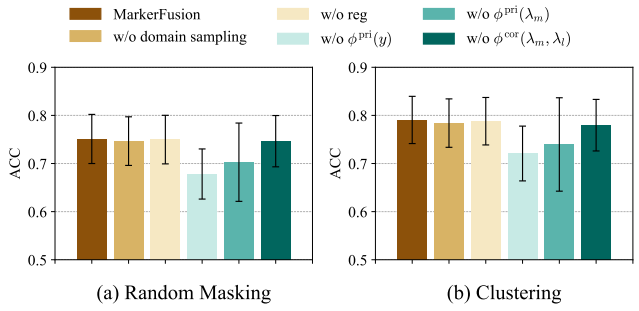


Figure 2: Ablation studies of key components. Scores are averaged across datasets with 95% confidence interval.

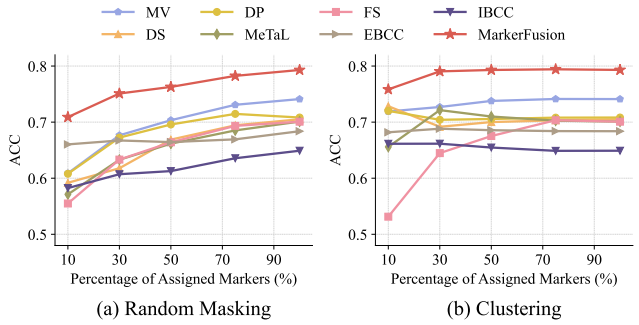


Figure 3: Accuracies w.r.t. percentage of assigned markers.

higher marker coverage rates after simulation. For completeness, we present the results for single-domain scenarios in Table 7 (Appendix D). MarkerFusion maintains its superior performance.

We note that it is a common phenomenon in established benchmarks [36, 66] that no single method can consistently outperform all others on every individual dataset, as different datasets underlie distinct distributions of data and latent variables. The compared methods, being unsupervised, make specific distributional assumptions that may align better with some datasets than others, leading to performance variations.

4.3 Ablation Study

To analyze the contribution of each component in our method, we conduct ablation experiments with five variants of MarkerFusion: (1) without domain sampling (denoted as *w/o domain sampling*), (2) without L1 regularization (denoted as *w/o reg*), (3) without the potential function for latent variable prior (denoted as $w/o\ \phi^{pri}(y)$), (4) without the potential function for marker prior (denoted as $w/o\ \phi^{pri}(\lambda_m)$), and (5) without the potential function for marker correlation (denoted as $w/o\ \phi^{cor}(\lambda_m, \lambda_l)$).

Figure 2 demonstrates that removing any component leads to performance degradation. This validates the effectiveness of our design choices. The removal of potential functions results in particularly significant accuracy drops. Besides the essential potential function $\phi(\lambda_m, y)$ that models marker accuracy, the learned priors for both the latent variable and markers show noticeable impact.

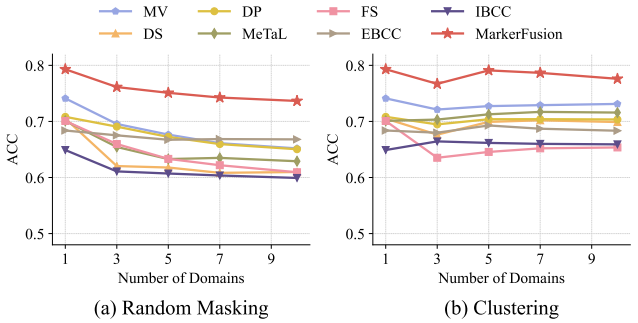


Figure 4: Accuracies w.r.t. number of domains.

4.4 Sensitivity Analyses

Percentage of assigned markers. We examine the effect of varying the percentage of markers per domain from {10%, 30%, 50%, 100%} while keeping the number of domains fixed at 5. Figure 3 shows the average accuracies across datasets. MarkerFusion always performs the best among all compared methods under different configurations. As expected, increasing the percentage of assigned markers per domain enhances prediction performance due to more comprehensive judgments from diverse sources. The AUC results in Figure 6 (Appendix D) show a consistent trend.

Number of domains. We change the number of domains from {1, 3, 5, 10} while maintaining the percentage of markers per domain at 30%. Figure 4 shows the average accuracies across datasets. MarkerFusion achieves the best performance under all configurations. For random masking, performance decreases with more domains since each domain contains fewer samples. For clustering-based domains, there is no clear trend due to the mixed effects of increasing domains: (1) more fine-grained clusters yield higher intra-domain marker coherence, potentially improving domain-specific modeling; (2) smaller sample size per domain and reduced correlated markers across domains limit cross-domain knowledge sharing. The AUC results in Figure 7 (Appendix D) show similar patterns.

4.5 Performance with Semi-Supervision

We evaluate MarkerFusion under different levels of supervision by varying the ratio of labeled samples from 0% (fully unsupervised) to 100% (fully supervised). We exclude the Spouse dataset from this analysis as it does not provide training labels. Figure 5 presents the accuracy across different supervision levels. Giving more labeled samples during training leads to improved accuracy and reduced variance (tighter 95% confidence intervals). The performance gains are moderate, suggesting that MarkerFusion can achieve robust performance even with little or no supervision.

5 Deployment and Impact

5.1 Deployment

MarkerFusion has been successfully deployed at global scale as a new detection capability of Amazon GuardDuty since December 2024. The deployment architecture consists of two primary components: a training pipeline and an inference pipeline, both designed

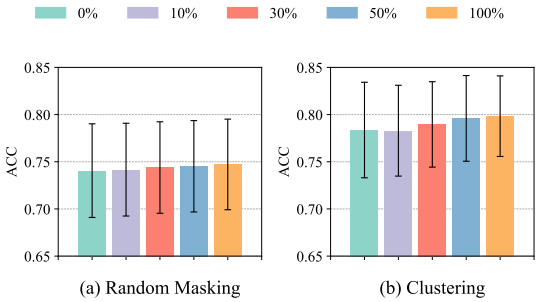


Figure 5: MarkerFusion with different ratios of labels.

Table 3: Summary of production datasets in four regions (AP: Asia Pacific, EU: Europe, US: United States).

Region	Collection Period	# Train	# Test
AP	Jul. 11 - Aug. 12, 2025	2,338,092	240,557
EU	Jun. 30 - Aug. 1, 2025	1,722,882	191,082
US West	Jun. 28 - Jul. 30, 2025	3,664,544	437,600
US East	Apr. 10 - May 12, 2025	1,268,358	80,904

for scalability, reliability, and continuous operation in cloud environments. The training pipeline leverages Amazon SageMaker to orchestrate model training and validation jobs. We implemented the data processing and model inference components as streaming applications using Apache Spark, deployed on Amazon Elastic MapReduce (EMR) clusters. This architecture enables near real-time threat detection while efficiently handling heterogeneous and high-volume data in the cloud environment. The system processes billions of alerts monthly from various AWS data sources and generates accurate attack findings that detect complex multi-stage attacks, which is a new extended capability of the detection system.

5.2 Performance on Production Data

Data collection. We assess model performance using data collected from four regions across continental zones, Asia Pacific (AP), Europe (EU), United States West and East. The statistics of the four datasets are listed in Table 3. The dataset consists of 4 domains. Each dataset spans approximately one month of collection with chronological training and testing splits to preserve temporal order.
Evaluation metrics. To evaluate whether the model aligns with insights from security experts, we utilize high-confidence attack patterns defined by internal security researchers as labels. Although these predefined patterns may introduce some noise into the evaluation process, they provide a scalable and reliable proxy for assessing model performance. Using these patterns as labels, we measure the AUC of pattern detection rates across two distinct datasets: external accounts experiencing real-world conditions, and internal accounts specifically configured for penetration testing that contain known attack behaviors. This allows us to assess the model's effectiveness in both production environments and controlled attack scenarios.
Experiment results. Table 4 presents the experiment results on our production data. MarkerFusion outperforms all compared

Table 4: AUC scores on collected production data.

Region	Scope	MV	DS	DP	MeTaL	FS	EBCC	IBCC	MarkerFusion
AP	Internal	0.895	0.503	0.908	0.504	0.874	0.610	0.901	**0.917**
	All	0.990	0.925	0.981	0.953	0.091	0.988	0.128	**0.998**
EU	Internal	0.895	0.502	0.890	0.504	0.865	0.719	0.847	**0.901**
	All	0.972	0.868	0.954	0.893	0.207	0.982	0.105	**0.985**
US East	Internal	0.919	0.541	0.933	0.541	0.817	0.759	0.806	**0.935**
	All	0.987	0.944	0.979	0.958	0.070	0.975	0.150	**0.997**
US West	Internal	0.859	0.996	0.723	**0.997**	0.858	0.722	0.858	**0.997**
	All	0.745	0.775	0.757	0.776	0.666	0.666	0.706	**0.922**

Table 5: Attack pattern hit rate of MarkerFusion with different number of labels in two data domains (US East).

Domain	0	20	60	100	200
A	0.788	0.788	0.804	0.804	0.946
B	0.998	0.998	0.998	0.998	0.998

methods across the four regions. It shows superior AUC on internal accounts and all accounts. Notably, MarkerFusion maintains robust performance across regions, while some baselines (e.g., FS, IBCC) show significant degradation in certain regions. The results above are purely unsupervised. Table 5 shows semi-supervised performance with MarkerFusion on two different data domains with the most data samples in the US East region, which provides sufficient labeled samples for training and evaluation. Domain A shows significant improvement from 0.788 to 0.946 with 200 labeled samples, demonstrating the effectiveness of incorporating supervision. Domain B maintains consistently high performance at 0.998, indicating that the unsupervised version already captures the underlying threat patterns effectively for this domain.

6 Conclusions

We present a novel probabilistic framework for aggregating predictions from multiple threat detectors across different cloud data domains. By conceptualizing individual detectors as markers with varying domain applicability, our approach effectively combines knowledge across different contexts. The framework naturally handles both unsupervised and semi-supervised settings through a principled Gibbs distribution model, allowing domain experts to contribute partial labels when available. Extensive experiments on multiple public datasets demonstrate that our method significantly outperforms established approaches in both multi-domain and single-domain scenarios. The framework's effectiveness is further validated through successful deployment at global scale as part of Amazon GuardDuty, where it processes billions of alerts monthly and identifies high-confidence attacks that would otherwise remain hidden. Our results demonstrate substantial improvements in detection accuracy for multi-stage attacks in cloud environments.

Acknowledgments

We thank the support of many people from Amazon GuardDuty team, in particular Aldrin Dsouza, Lucas Winkelmann, Jeffery Bickford, and Baris Coskun.

References

[1] 1981. Mushroom. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5959T.

[2] Bikash Agrawal, Tomasz Wiktorski, and Chunming Rong. 2017. Adaptive real-time anomaly detection in cloud infrastructures. *Concurrency and Computation: Practice and Experience* 29, 24 (2017), e4193.

[3] Siraaj Akhtar, Saad Khan, and Simon Parkinson. 2025. LLM-based event log analysis techniques: A survey. *arXiv preprint arXiv:2502.00677* (2025).

[4] Túlio C Alberto, Johannes V Lochter, and Tiago A Almeida. 2015. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*. IEEE, 138–143.

[5] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of SMS spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*. 259–262.

[6] K Arivarasan and Mohammad S Obaidat. 2022. Intrusion Detection System using Aggregation of Machine Learning Algorithms. In *2022 International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE, 1–8.

[7] Stephen H Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning*. PMLR, 273–282.

[8] Jay Barach. 2025. AI-Driven Causal Inference for Cross-Cloud Threat Detection Using Anonymized CloudTrail Logs. In *2025 Conference on Artificial Intelligence x Multimedia (AIxMM)*. IEEE, 45–50.

[9] Sean Barnum. 2012. Standardizing cyber threat intelligence information with the structured threat information expression (stix). *Mitre Corporation* 11, 2012 (2012), 1–22.

[10] Tobias Braun, Irdin Pekaric, and Giovanni Apruzzese. 2024. Understanding the process of data labeling in cybersecurity. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*. 1596–1605.

[11] Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. 2021. Dependency structure misspecification in multi-source weak supervision models. *arXiv preprint arXiv:2106.10302* (2021).

[12] George Casella and Edward I George. 1992. Explaining the Gibbs sampler. *The American Statistician* 46, 3 (1992), 167–174.

[13] Onur Catakoglu, Marco Balduzzi, and Davide Balzarotti. 2016. Automatic extraction of indicators of compromise for web applications. In *Proceedings of the 25th international conference on world wide web*. 333–343.

[14] Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. 2020. Robust data programming with precision-guided labeling functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3397–3404.

[15] David PA Corney, Dyaa Albakour, Miguel Martinez-Alvarez, and Samir Moussa. 2016. What do a million news articles look like?. In *NewsIR@ ECIR*. 42–47.

[16] Allan Peter Davis, Cynthia J Grondin, Robin J Johnson, Daniela Sciaky, Benjamin L King, Roy McMorran, Jolene Wiegers, Thomas C Wiegers, and Carolyn J Mattingly. 2017. The comparative toxicogenomics database: update 2017. *Nucleic acids research* 45, D1 (2017), D972–D978.

[17] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28, 1 (1979), 20–28.

[18] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)* 39, 1 (1977), 1–22.

[19] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.

[20] Marius Dragoi, Elena Burceanu, Emanuela Haller, Andrei Manolache, and Florin Brad. 2022. Anoshift: A distribution shift benchmark for unsupervised anomaly detection. *Advances in Neural Information Processing Systems* 35 (2022), 32854–32867.

[21] Daniel Fraunholz, Daniel Schneider, Janis Zemitis, and Hans Dieter Schotten. 2018. Hack my company: An empirical assessment of post-exploitation behavior and lateral movement in cloud environments. In *Proceedings of the central European cybersecurity conference 2018*. 1–6.

[22] Daniel Fu, Mayee Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International conference on machine learning*. PMLR, 3280–3291.

[23] Dr A SHAJI GEORGE, AS Hovan George, Thangaraj Baskar, and Digvijay Pandey. 2021. XDR: the evolution of endpoint security solutions-superior extensibility and analytics to satisfy the organizational needs of the future. *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)* 8, 1 (2021), 493–501.

[24] Ben Hamner, David C Thompson, and Joerg Bentzien. 2012. Predicting a biological response. Kaggle competition. https://www.kaggle.com/competitions/bioresponse

[25] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525* (2020).

[26] FC Heilbron, V Escorcia, B Ghanem, and J Niebles. 2019. A largescale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, 2015. 961*, Vol. 970.

[27] Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. 1999. Spambase. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C53G6X.

[28] Zhexue Huang et al. 1997. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD)*. Citeseer, 21–34.

[29] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*. 64–67.

[30] Hanan Ghali Jabbar. 2024. Advanced threat detection using soft and hard voting techniques in ensemble learning. *Journal of Robotics and Control (JRC)* 5, 4 (2024), 1104–1116.

[31] Ariel Jaffe, Ethan Fetaya, Boaz Nadler, Tingting Jiang, and Yuval Kluger. 2016. Unsupervised ensemble learning with dependent classifiers. In *Artificial Intelligence and Statistics*. PMLR, 351–360.

[32] Robert J Joyce, Derek Everett, Maya Fuchs, Edward Raff, and James Holt. 2025. Claravy: A tool for scalable and accurate malware family labeling. In *Companion Proceedings of the ACM on Web Conference 2025*. 277–286.

[33] Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *Artificial Intelligence and Statistics*. PMLR, 619–627.

[34] Ron Kohavi et al. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.. In *Kdd*, Vol. 96. 202–207.

[35] Heng Li, Shiyao Zhou, Wei Yuan, Xiapu Luo, Cuiying Gao, and Shuiyan Chen. 2021. Robust android malware detection against adversarial example attacks. In *Proceedings of the Web Conference 2021*. 3603–3612.

[36] Yuan Li, Benjamin Rubinstein, and Trevor Cohn. 2019. Exploiting worker correlation for label aggregation in crowdsourcing. In *International conference on machine learning*. PMLR, 3886–3895.

[37] Po-Ling Loh and Martin J Wainwright. 2012. Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. *Advances in Neural Information Processing Systems* 25 (2012).

[38] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 142–150.

[39] Simon Miller, Christian Wagner, Uwe Aickelin, and Jonathan M Garibaldi. 2016. Modelling cyber-security experts' decision making processes using aggregation operators. *computers & security* 62 (2016), 229–245.

[40] Sérgio Moro, Paulo Cortez, and Paulo Rita. 2014. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems* 62 (2014), 22–31.

[41] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, Halah Albadani, and Fatima Mohamad Dakalbab. 2021. Machine learning for cloud security: a systematic review. *IEEE Access* 9 (2021), 20717–20735.

[42] Arvind Prasad and Shalini Chandra. 2024. PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. *Computers & Security* 136 (2024), 103545.

[43] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB endowment. International conference on very large data bases*, Vol. 11. 269.

[44] Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4763–4771.

[45] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems* 29 (2016).

[46] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of machine learning research* 11, 4 (2010).

[47] Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie Mitchell, and Chao Zhang. 2020. Denoising multi-source weak supervision for neural text classification. *arXiv preprint arXiv:2010.04582* (2020).

[48] Mir Shahnawaz Ahmad and Shahid Mehraj Shah. 2022. Unsupervised ensemble based deep learning approach for attack detection in IoT network. *Concurrency and Computation: Practice and Experience* 34, 27 (2022), e7338.

[49] Edwin Simpson, Stephen Roberts, Ioannis Psorakis, and Arfon Smith. 2013. Dynamic bayesian combination of multiple imperfect classifiers. In *Decision making and imperfection*. Springer, 1–35.

[50] Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. 1994. Inferring ground truth from subjective labelling of venus images. *Advances in neural information processing systems* 7 (1994).

[51] Bhavna Soman, Ali Torkamani, Michael J Morais, Jeffrey Bickford, and Baris Coskun. 2022. Firenze: Model evaluation using weak signals. *arXiv preprint arXiv:2207.00827* (2022).

[52] Nan Sun, Ming Ding, Jiaojiao Jiang, Weikang Xu, Xiaoxing Mo, Yonghang Tai, and Jun Zhang. 2023. Cyber threat intelligence mining for proactive cybersecurity defense: A survey and new perspectives. *IEEE Communications Surveys & Tutorials* 25, 3 (2023), 1748–1774.

[53] Hamed Tabrizchi and Marjan Kuchaki Rafsanjani. 2020. A survey on security challenges in cloud computing: issues, threats, and solutions. *The journal of supercomputing* 76, 12 (2020), 9493–9532.

[54] Saravanan Thirumuruganathan, Mohamed Nabeel, Euijin Choo, Issa Khalil, and Ting Yu. 2022. Siraj: A unified framework for aggregation of malicious entity detectors. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 507–521.

[55] Ke Tian, Steve TK Jan, Hang Hu, Danfeng Yao, and Gang Wang. 2018. Needle in a haystack: Tracking down elite phishing domains in the wild. In *Proceedings of the Internet Measurement Conference 2018*. 429–442.

[56] Alexandra N Uma, Tommaso Fornaciari, Dirk Hovy, Silviu Paun, Barbara Plank, and Massimo Poesio. 2021. Learning from disagreement: A survey. *Journal of Artificial Intelligence Research* 72 (2021), 1385–1470.

[57] Blesson Varghese and Rajkumar Buyya. 2018. Next generation cloud computing: New trends and research directions. *Future generation computer systems* 79 (2018), 849–861.

[58] Paroma Varma, Frederic Sala, Ann He, Alexander Ratner, and Christopher Ré. 2019. Learning dependency structures for weak supervision models. In *International Conference on Machine Learning*. PMLR, 6418–6427.

[59] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier Movellan, and Paul Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in neural information processing systems* 22 (2009).

[60] Yanqing Yang, Kangfeng Zheng, Chunhua Wu, Xinxin Niu, and Yixian Yang. 2019. Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks. *Applied Sciences* 9, 2 (2019), 238.

[61] Peilin Yu, Tiffany Ding, and Stephen H Bach. 2022. Learning from multiple noisy partial labelers. In *International conference on artificial intelligence and statistics*. PMLR, 11072–11095.

[62] Jing Zhang. 2022. Knowledge learning with crowdsourcing: A brief review and systematic perspective. *IEEE/CAA Journal of Automatica Sinica* 9, 5 (2022), 749–762.

[63] Jie Zhang, Haoyu Bu, Hui Wen, Yu Chen, Lun Li, and Hongsong Zhu. 2024. When LLMs meet cybersecurity: a systematic literature review (2024). *arXiv preprint arXiv:2405.03644* (2024).

[64] Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. 2022. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433* (2022).

[65] Jiayun Zhang, Junshen Xu, Bugra Can, and Yi Fan. 2025. React: Residual-adaptive contextual tuning for fast model adaptation in threat detection. In *Proceedings of the ACM on Web Conference 2025*. 2488–2499.

[66] Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. WRENCH: A Comprehensive Benchmark for Weak Supervision. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=Q9SKS5k8io

A Symbols and Definitions

Table 6 lists the notations used in this paper. We use bold letters for vectors, plain for scalars, capital letters for random variables, and lower-case for realizations.

Symbol	Definition
Y	ground truth variable
y	value of the ground truth
S	data domain variable
s	value of data domain
$\boldsymbol{\Lambda} = [\Lambda_1, \ldots, \Lambda_M]$	vector of markers
$\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_M]$	observation of marker vector
M	number of markers
N	number of samples
$\boldsymbol{\theta} = [\theta_i, \ldots, \theta_K]$	parameter vector
$\boldsymbol{\phi} = [\phi_1, \ldots, \phi_K]$	vector of potential functions
K	number of parameters/potential functions
p_X	distribution of random variable X

Table 6: Symbols and definitions

B Derivation of Gradient

B.1 Unlabeled Data

To derive the gradient in equation (3), we consider the gradient for a sample i without label.

$$
\begin{aligned}
&\frac{\partial \log p_{\boldsymbol{\Lambda}|S}(\boldsymbol{\lambda}^{(i)}|s^{(i)};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \\
&= \frac{\partial}{\partial \boldsymbol{\theta}} \log \sum_y p_{\boldsymbol{\Lambda},Y|S}(\boldsymbol{\lambda}^{(i)}, y|s^{(i)};\boldsymbol{\theta}) \\
&= \frac{\partial}{\partial \boldsymbol{\theta}} \log \sum_y \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y, s^{(i)})\right) \\
&\quad - \frac{\partial}{\partial \boldsymbol{\theta}} \log Z(s^{(i)}, \boldsymbol{\theta}) \\
&= \frac{\sum_y \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y, s^{(i)})\right) \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y, s^{(i)})}{\sum_y \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y, s^{(i)})\right)} \\
&\quad - \frac{1}{Z(s^{(i)}, \boldsymbol{\theta})} \sum_{\boldsymbol{\lambda}', y'} \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{\lambda}', y', s^{(i)})\right) \boldsymbol{\phi}(\boldsymbol{\lambda}', y', s^{(i)}) \\
&= \sum_y p_{Y|\boldsymbol{\Lambda},S}(y|\boldsymbol{\lambda}^{(i)}, s^{(i)}) \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y, s^{(i)}) \\
&\quad - \sum_{\boldsymbol{\lambda}', y'} p_{\boldsymbol{\Lambda},Y|S}(\boldsymbol{\lambda}', y', s^{(i)}) \boldsymbol{\phi}(\boldsymbol{\lambda}', y', s^{(i)}) \\
&= \mathbb{E}_{p_{Y|\boldsymbol{\Lambda}=\boldsymbol{\lambda}^{(i)}, S=s^{(i)}}} \boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, Y, s^{(i)}) - \mathbb{E}_{p_{\boldsymbol{\Lambda},Y|S=s^{(i)}}} \boldsymbol{\phi}(\boldsymbol{\Lambda}, Y, s^{(i)}).
\end{aligned}
$$

We complete the proof by taking average over all samples in the dataset.

B.2 Data with Hard Label

The gradient in equation (5) consists of two parts, unlabeled data from $\mathcal{A}$ and labeled data from $\mathcal{B}$. The gradient of unlabeled data is the same as equation (3). For the gradient of labeled data, we consider the gradient for a sample i with hard label $y^{(i)}$.

$$
\begin{aligned}
&\frac{\partial \log p_{\mathbf{\Lambda},Y|S}(\boldsymbol{\lambda}^{(i)}, y^{(i)}|s^{(i)};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \\
=&\frac{\partial}{\partial \boldsymbol{\theta}}\left[\boldsymbol{\theta}^T\boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y^{(i)}, s^{(i)}) - \log Z(s^{(i)},\boldsymbol{\theta})\right] \\
=&\boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y^{(i)}, s^{(i)}) - \frac{1}{Z(s^{(i)},\boldsymbol{\theta})}\sum_{\boldsymbol{\lambda}',y'}\exp\left(\boldsymbol{\theta}^T\boldsymbol{\phi}(\boldsymbol{\lambda}', y', s^{(i)})\right)\boldsymbol{\phi}(\boldsymbol{\lambda}', y', s^{(i)}) \\
=&\boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y^{(i)}, s^{(i)}) - \sum_{\boldsymbol{\lambda}',y'} p_{\mathbf{\Lambda},Y|S}(\boldsymbol{\lambda}', y', s^{(i)})\boldsymbol{\phi}(\boldsymbol{\lambda}', y', s^{(i)}) \\
=&\boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y^{(i)}, s^{(i)}) - \mathbb{E}_{p_{\mathbf{\Lambda},Y|S=s^{(i)}}}\boldsymbol{\phi}(\mathbf{\Lambda}, Y, s^{(i)}).
\end{aligned}
$$

B.3 Data with Soft Label

In the cases where we only have access to the ground truth label of Y with some uncertainty (referred to as soft label), we further modify the objective function in Equation 5 to incorporate the uncertainty. Let $q_Y^{(i)}(y)$ be the soft label for sample i, where $q_Y^{(i)}(y) \geq 0, \forall y \in \mathcal{Y}$ ($\mathcal{Y}$ is the alphabet of Y) and $\sum_{y\in\mathcal{Y}} q_Y^{(i)}(y) = 1$. In the case of soft label, the second term in equation (5) can be generalized to the cross entropy between the soft label $q_Y^{(i)}(\cdot)$ and $p_{\mathbf{\Lambda},Y}(\boldsymbol{\lambda}^{(i)}, \cdot;\boldsymbol{\theta})$, i.e.,

$$
\begin{aligned}
\mathcal{L}^{\text{soft}}_{\text{semi}}(\boldsymbol{\theta}) = &\frac{1}{N}\sum_{i\in\mathcal{A}}\log p_{\mathbf{\Lambda}|S}(\boldsymbol{\lambda}^{(i)}|s^{(i)};\boldsymbol{\theta}) \\
&+\frac{1}{N}\sum_{i\in\mathcal{B}}\sum_{y\in\mathcal{Y}} q_Y^{(i)}(y)\log p_{\mathbf{\Lambda},Y|S}(\boldsymbol{\lambda}^{(i)}, y|s^{(i)};\boldsymbol{\theta}).
\end{aligned} \tag{7}
$$

We can see equation (4) as a special case of equation (7) by setting $q_Y^{(i)} = \mathbb{I}(y = y^{(i)})$. The gradient of $\mathcal{L}^{\text{soft}}_{\text{semi}}$ with respect to $\boldsymbol{\theta}$ can be computed by

$$
\begin{aligned}
\frac{\partial \mathcal{L}^{\text{soft}}_{\text{semi}}}{\partial \boldsymbol{\theta}} = &\frac{1}{N}\sum_{i\in\mathcal{A}}\mathbb{E}_{p_{Y|\mathbf{\Lambda}=\boldsymbol{\lambda}^{(i)},S=s^{(i)}}}\boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, Y, s^{(i)}) \\
&+\frac{1}{N}\sum_{j\in\mathcal{B}}\mathbb{E}_{q_Y^{(i)}}\boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, Y, s^{(i)}) \\
&-\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{p_{\mathbf{\Lambda},Y|S=s^{(i)}}}\boldsymbol{\phi}(\mathbf{\Lambda}, Y, s^{(i)}).
\end{aligned} \tag{8}
$$

To derive the gradient in equation (8), we consider the gradient for a sample i with soft label $q_Y^{(i)}$.

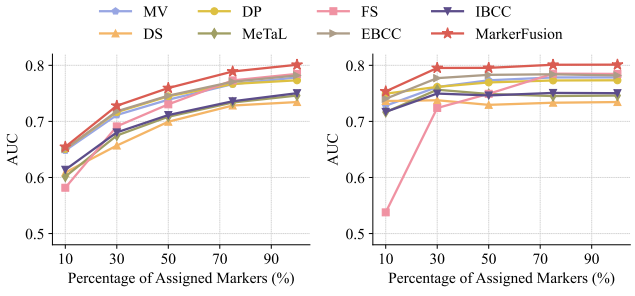
$$
\begin{aligned}
&\frac{\partial \sum_{y\in\mathcal{Y}} q_Y^{(i)}(y)\log p_{\mathbf{\Lambda},Y|S}(\boldsymbol{\lambda}^{(i)}, y, s^{(i)})}{\partial \boldsymbol{\theta}} \\
=&\frac{\partial}{\partial \boldsymbol{\theta}}\left[\sum_{y\in\mathcal{Y}} q_Y^{(i)}(y)\boldsymbol{\theta}^T\boldsymbol{\phi}(\boldsymbol{\lambda}^{(i)}, y, s^{(i)}) - \log Z(s^{(i)},\boldsymbol{\theta})\right] \\
=&\sum_{y\in\mathcal{Y}} q_Y^{(i)}(y)\phi(\boldsymbol{\lambda}^{(i)}, y, s^{(i)}) \\
&-\frac{1}{Z(s^{(i)},\boldsymbol{\theta})}\sum_{\boldsymbol{\lambda}',y'}\exp\left(\boldsymbol{\theta}^T\boldsymbol{\phi}(\boldsymbol{\lambda}', y', s^{(i)})\right)\boldsymbol{\phi}(\boldsymbol{\lambda}', y', s^{(i)}) \\
=&\sum_{y\in\mathcal{Y}} q_Y^{(i)}(y)\phi(\boldsymbol{\lambda}^{(i)}, y, s^{(i)}) - \sum_{\boldsymbol{\lambda}',y'} p_{\mathbf{\Lambda},Y|S}(\boldsymbol{\lambda}', y'|s^{(i)})\boldsymbol{\phi}(\boldsymbol{\lambda}', y', s^{(i)}) \\
=&\mathbb{E}_{q_Y^{(i)}}\phi(\boldsymbol{\lambda}^{(i)}, Y, s^{(i)}) - \mathbb{E}_{p_{\mathbf{\Lambda},Y|S=s^{(i)}}}\boldsymbol{\phi}(\mathbf{\Lambda}, Y, s^{(i)}).
\end{aligned}
$$

C Ranking

$$
\begin{aligned}
r(\boldsymbol{\lambda}, s;\boldsymbol{\theta}) &= \boldsymbol{\theta}^{\mathsf{T}}(\phi(\boldsymbol{\lambda}, 1, s) - \phi(\boldsymbol{\lambda}, -1, s)) \\
&= \log\frac{\exp(\theta^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{\lambda}, 1, s))}{\exp(\theta^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{\lambda}, -1, s))} \\
&= \log\frac{p_{\mathbf{\Lambda},Y|S}(\boldsymbol{\lambda}, 1|s;\boldsymbol{\theta})}{p_{\mathbf{\Lambda},Y|S}(\boldsymbol{\lambda}, -1|s;\boldsymbol{\theta})} \\
&= \log\frac{p_{Y|\mathbf{\Lambda},S}(1|\boldsymbol{\lambda}, s;\boldsymbol{\theta})p_{\mathbf{\Lambda}|S}(\boldsymbol{\lambda}|s;\boldsymbol{\theta})}{p_{Y|\mathbf{\Lambda},S}(-1|\boldsymbol{\lambda}, s;\boldsymbol{\theta})p_{\mathbf{\Lambda}|S}(\boldsymbol{\lambda}|s;\boldsymbol{\theta})} \\
&= \log\frac{p_{Y|\mathbf{\Lambda},S}(1|\boldsymbol{\lambda}, s;\boldsymbol{\theta})}{p_{Y|\mathbf{\Lambda},S}(-1|\boldsymbol{\lambda}, s;\boldsymbol{\theta})} \\
&= \log\frac{p_{Y|\mathbf{\Lambda},S}(1|\boldsymbol{\lambda}, s;\boldsymbol{\theta})}{1 - p_{Y|\mathbf{\Lambda},S}(1|\boldsymbol{\lambda}, s;\boldsymbol{\theta})} \\
&= g(p_{Y|\mathbf{\Lambda},S}(1|\boldsymbol{\lambda}, s;\boldsymbol{\theta})),
\end{aligned}
$$

where $g(t) = \log(t/(1-t))$. Since $g(t)$ is monotonically increasing when $t \in (0, 1)$, ranking by score $r(\boldsymbol{\lambda}, s;\boldsymbol{\theta})$ is equivalent to ranking by posterior $p_{Y|\mathbf{\Lambda},S}(1|\boldsymbol{\lambda}, s;\boldsymbol{\theta})$.

D Additional Experiment Results



(a) Random Masking (b) Clustering

Figure 6: AUC scores w.r.t. percentage of assigned markers.

Figure 6 and Figure 7 show the AUC scores under different configurations for the number of domains and the percentage of assigned markers per domain. The results conform with the analyses we

Table 7: Experiment results on public datasets with single-domain scenarios

Dataset	MV		DS		DP		MeTaL		FS		EBCC		IBCC		MarkerFusion	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
YouTube	0.840	0.905	**0.848**	0.892	0.776	0.884	0.544	0.668	0.788	0.866	0.528	0.836	0.528	0.766	0.816	**0.927**
Census	0.799	0.734	0.480	0.692	0.776	0.775	0.696	0.737	0.779	0.822	0.764	0.835	0.764	0.793	**0.814**	**0.863**
SMS	0.632	0.521	0.836	0.500	0.630	0.520	0.612	0.508	0.500	0.507	0.844	**0.524**	**0.864**	0.519	0.844	0.519
IMDb	0.710	0.752	0.712	0.764	0.710	0.763	0.704	0.736	0.703	0.758	**0.714**	0.765	0.548	0.763	0.709	**0.766**
Yelp	0.702	0.766	0.715	0.794	0.694	0.780	0.553	0.720	0.715	**0.807**	0.653	0.804	0.634	0.791	**0.715**	0.800
Spouse	0.521	0.767	0.187	0.281	0.529	0.763	0.521	0.752	0.518	0.779	0.186	0.775	0.081	**0.786**	**0.884**	0.754
CDR	0.694	0.787	0.396	0.665	0.644	0.611	0.641	0.640	**0.708**	0.778	0.682	0.690	0.676	0.539	0.643	**0.798**
Commercial	0.902	0.963	0.892	0.968	0.821	0.968	0.906	0.966	0.869	0.966	0.821	0.967	0.824	0.967	**0.925**	**0.974**
Tennis Rally	0.862	0.882	0.857	0.876	**0.881**	0.884	0.859	**0.886**	0.853	0.885	0.875	0.884	0.874	0.884	0.861	0.885
Basketball	0.698	0.529	0.857	0.529	0.549	0.527	**0.899**	0.529	0.468	0.500	0.859	0.527	0.886	0.527	0.860	**0.530**
Bank Marketing	0.723	0.820	0.770	0.838	0.807	**0.843**	0.808	0.828	0.674	0.815	**0.889**	0.832	**0.889**	0.825	0.823	0.840
Bioresponse	0.617	0.673	**0.668**	0.696	0.519	0.663	0.535	0.692	0.551	0.685	0.521	0.683	0.521	0.670	0.630	**0.701**
Mushroom	0.870	**0.965**	0.862	0.873	0.865	0.946	0.871	0.927	**0.881**	0.936	0.738	0.901	0.471	0.872	0.871	0.932
PhiUSIIL	0.772	**0.861**	0.732	0.741	0.709	0.773	0.680	0.699	0.769	0.845	0.570	0.821	0.570	0.683	**0.779**	0.835
Spambase	**0.774**	0.751	0.768	**0.911**	0.712	0.854	0.705	0.879	0.740	0.852	0.605	0.892	0.605	0.871	0.720	0.894
Average	0.741	0.778	0.705	0.735	0.708	0.770	0.702	0.744	0.701	0.787	0.683	0.782	0.649	0.750	**0.793**	**0.801**
Mean Rank	3.4	4.9	4.4	4.7	4.9	4.5	5.1	5.6	4.7	4.5	4.8	3.7	5.3	5.5	**2.7**	**2.3**

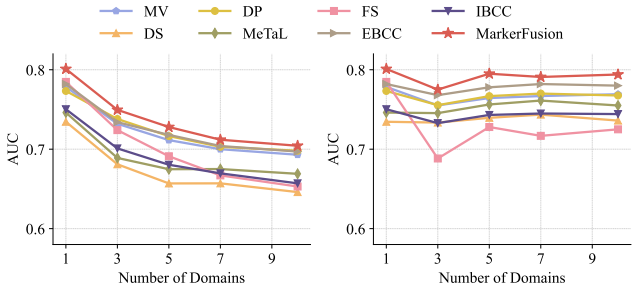


Figure 7: AUC scores w.r.t. number of domains.

presented earlier in Section 4.4. It shows the same tendency as the accuracies—the AUC scores increase as the percentage of assigned markers per domain grows, while increasing the number of domains leads to slightly degraded performance with random masking-based multi-domains due to reduced sample sizes. This consistent pattern reinforces our previous findings.

Table 7 presents results for single-domain scenarios. While performance gaps between methods are smaller than in multi-domain scenarios, MarkerFusion still achieves the highest average accuracy and AUC among all compared methods.

E Efficiency in Training and Inference

We evaluate the computational efficiency of MarkerFusion in terms of both training time and inference latency. Training the MarkerFusion model on the dataset described in Section 5.2 requires approximately 17 minutes using an instance equipped with 96 vCPUs. At inference time, the model processes each event in 5.3 ms per core, enabling real-time detection in streaming applications.