CARE: Certifiably Robust Learning with Reasoning via Variational Inference

***Abstract*—Despite great recent advances achieved by deep neural networks (DNNs), they are often vulnerable to adversarial attacks. Intensive research efforts have been made to improve the robustness of DNNs; however, most *empirical* defenses can be adaptively attacked again, and the *theoretically* certified robustness is limited, especially on large-scale datasets. One potential root cause of such vulnerabilities for DNNs is that although they have demonstrated powerful expressiveness, they lack the reasoning ability to make robust and reliable predictions. In this paper, we aim to integrate domain knowledge to enable robust learning with the *reasoning* paradigm. In particular, we propose a certifiably robust learning with reasoning pipeline (CARE), which consists of a *learning* component and a *reasoning* component. Concretely, we use a set of standard DNNs to serve as the learning component to make semantic predictions (e.g., whether the input is furry), and we leverage the probabilistic graphical models, such as Markov logic networks (MLN), to serve as the reasoning component to enable knowledge/logic reasoning (e.g., `IsPanda` $\implies$ `IsFurry`). However, it is known that the exact inference of MLN (reasoning) is #P-complete, which limits the scalability of the pipeline. To this end, we propose to approximate the MLN inference via variational inference based on an efficient expectation maximization algorithm. In particular, we leverage graph convolutional networks (GCNs) to encode the posterior distribution during variational inference and update the parameters of GCNs (E-step) and the weights of knowledge rules in MLN (M-step) iteratively. We conduct extensive experiments on different datasets such as AwA2, Word50, GTSRB, and PDF malware, and we show that CARE achieves significantly higher certified robustness (e.g., the certified accuracy is improved from 36.0% to 61.8% under ℓ_2 radius 2.0 on AwA2) compared with the state-of-the-art baselines. We additionally conducted different ablation studies to demonstrate the empirical robustness of CARE and the effectiveness of different knowledge integration.**

***Index Terms*—Robust learning with reasoning, Markov logic network, graph convolutional network, certified robustness, variational inference.**

I. Introduction

Despite that machine learning (ML), especially deep neural networks (DNNs), have achieved great successes in different applications, they are also found to be vulnerable to small and adversarial perturbations that could lead to incorrect predictions [1]–[4]. Given the massive deployment of machine learning systems, especially in safety-critical scenarios such as automatic driving [5], [6] and medical diagnosis [7], [8], improving the robustness of ML models is of great importance, and a reliable defense mechanism is in dire need.

To overcome such adversarial attacks, significant efforts have been made to develop different defense approaches, both empirically and theoretically [9]–[13]. However, most of these existing empirical defenses have been attacked successfully again by strong adaptive attacks [14], [15]; and the theoretically certifiably robust models are usually limited on large-scale data [16]–[19]. On the other hand, existing ML models lack logical reasoning abilities, which may be one main root cause of their vulnerabilities. For instance, a human would be able to recognize a stop sign by just seeing the octagon shape, while DNNs cannot reason based on such knowledge. Thus, in this paper, we aim to explore the question: *Can we integrate domain knowledge into statistical learning with DNNs to improve their robustness? Will the certified robustness of ML models be improved when composed with a reasoning component? Can we do such integration in an efficient and scalable way?*

To effectively integrate knowledge rules to enable reasoning ability for existing DNN-based statistical learning, in this work, we propose a *learning with reasoning* pipeline CARE, which contains both a *learning* and a *reasoning* component. In particular, the learning component contains one *main sensor* that is in charge of the main classification task (e.g., d-way animal prediction) and several *knowledge sensors* that identify different semantic entities or attributes (e.g., whether the input is furry). The output of different sensors will be taken as the input of the *reasoning* component, which can be realized using probabilistic graphical models such as Markov logic networks (MLN) [20]. Concretely, different knowledge rules (e.g., "Panda is furry") can be represented as the first-order logic rules and then embedded in the MLN to help perform logic reasoning. The overall pipeline of CARE is shown in Figure 1. The advantage of such a pipeline is that the predictions of different sensors are dependent, following the logical relationships among them. Thus, given an attack against, say, the main sensor, the adversary not only needs to attack a set of sensors additionally but also needs to ensure that the attacked predictions of these sensors satisfy the logical relationships, making the attack much more challenging in practice.

Although such reasoning integration is very promising, as illustrated in a few recent seminal explorations [21], [22], scalability and efficiency hinder their real-world applications — the inference of MLN is #P-complete [20], which is exponential in the number of the possible predictions of sensors within the logical relationships, and thus impedes the scalability of such pipelines. As a result, recent attempts in using reasoning to improve robustness can only handle relatively small-scale problems [22]. The key technical contribution of this paper is to approximate the MLN inference with variational inference based on parametrized graphical convolutional networks (GCN) [23]. Currently, in addition to the classical

inference approximation such as Markov chain Monte Carlo (MCMC) [24], [25], and loopy belief propagation [26], variational method [27] which approximates probability densities through optimization has become more and more efficient and convenient in practice owing to advanced learning strategies. On a high level, the variational method approximates the posterior distribution with a given approximating function family $\mathcal{Q}$, and thus the design of such function family largely affects the final approximation. More specifically, this function family should satisfy the following two requirements: (1) it should capture the topology of the knowledge/logic relationship structure; (2) it should be scalable and can be optimized effectively on large-scale datasets. To this end, we follow the observations of existing works [28], [29] and adopt GCN to serve as the approximating function, which can efficiently represent the large knowledge graph structure.

Concretely, we will map each sensor prediction (e.g., Panda) as a node within GCN and the logical relationships between sensors as edges (e.g., an edge between sensors predicting "Panda" and "furry" to represent the rule `IsPanda(x)` $\implies$ `IsFurry(x)`). Since the inference of GCN scales linearly in the number of graph edges, the corresponding approximated inference of MLNs can thus scale linearly in the number of knowledge rules, which makes the CARE scalable to large-scale problems. In particular, we propose an efficient expectation maximization algorithm to iteratively update the weights of GCN (E-step) and the weights of logic rules within MLN (M-step).

This allows us to apply CARE to an unprecedented scale on problems that involve logical reasoning to improve robustness. To demonstrate the robustness of CARE, we conduct extensive experiments on four large-scale datasets: Animals with Attributes (AwA2) [30], Word50 [31], GTSRB [32], and PDF malware dataset from Contagio [33]. For AwA2, we leverage the annotated attributes of each animal class and the hierarchy relationship among different animal categories extracted from WordNet [34] as our knowledge rules. For Word50, we leverage the positions of each character in the known words as knowledge rules. For GTSRB, we use the road sign properties such as shape and the content of each sign as knowledge rules; while for the PDF malware dataset, the common benign/malicious traces (features) (e.g., `/Root/Pages/Contents/Filter` for benign trace and `/Root/OpenAction` for malicious trace) and their relationships are used to construct the knowledge rules. We show that CARE significantly outperforms the SOTA *certified* defenses [35]–[38] under different perturbation radii. We also conduct different ablation studies to further understand the impacts of the number of integrated knowledge rules, the empirical robustness of CARE, and the robustness and explanation properties of CARE based on case studies.

Technical Contributions. In this paper, we provide a scalable certifiably robust *learning with reasoning* pipeline CARE, which has demonstrated significantly higher certified robustness than baselines on large-scale image datasets as well as PDF malware dataset.

- We propose a scalable and certifiably robust *learning with reasoning* pipeline CARE, which is able to integrate knowledge rules to enable reasoning ability for reliable prediction.
- We propose an efficient expectation maximization algorithm to approximate the reasoning (MLN) inference via variational inference using GCN.
- We conduct extensive experiments on a wide range of datasets and demonstrate that CARE achieves *significantly* higher certified robustness than SOTA baselines. For instance, CARE improves the certified accuracy from 36.0% (SOTA) to 61.8% under ℓ_2 radius 2.0 on AwA2; and for the word-level classification on Word50, CARE improves the certified accuracy from 24.8% (SOTA) to 73.6% under ℓ_2 radius 0.5; on the GTSRB dataset, CARE improves the certified accuracy from 83.3% (SOTA) to 84.4% under ℓ_2 radius 0.4; for PDF malware, CARE improves the certified accuracy from 22.6% (SOTA) to 54.5% under ℓ_0 radius 7.
- We conduct a set of ablation studies to explore the impact of the number of integrated knowledge rules; demonstrate the high empirical robustness of CARE compared with baselines; and showcase the robustness and explanation properties of CARE based on case studies.

II. Background

Markov Logic Networks (MLN) provides an effective approach to combining first-order logic and probabilistic graphical models in a unified representation. Concretely, MLN can be viewed as a first-order knowledge base with a weight attached to each logic formula, where the first-order logic formula can be used to model different types of domain knowledge such as "`IsPanda(x)` $\implies$ `IsFurry(x)`". Formally, in MLN, the mapping (prediction) among entities can be represented as *predicates* $t(\cdot)$, which is a logic function defined over the entity variable set $\mathcal{V} = \{v_1, ..., v_N\}$ where v_i denotes a *constant* in the logic world. For instance, a constant can be a "stop sign" or a "octagon shape". The predicate is thus defined as $t(\cdot) : \mathcal{V} \times ... \times \mathcal{V} \to \{0, 1\}$. In the meantime, a logic *formula* in MLN is defined over the composition of a set of predicates as $f(\cdot) : \mathcal{V} \times ... \times \mathcal{V} \to \{0, 1\}$. For instance, given an input instance x, a model that is trained to predict whether the input is of the octagon shape $g_{octagon}(x)$ can be viewed as a predicate. The related knowledge rules, such as "stop sign $\implies$ octagon shape" can be represented as a formula as

$$g_{stop}(x) \implies g_{octagon}(x).$$

A formula consists of different predicates. We denote the assignments of variables to the arguments of a formula f as a_f, and all the possible consistent assignments are represented as set $\mathcal{A}_f = \{a_f^1, a_f^2, ...\}$. With a particular set of constants assigned to the arguments of a predicate, it is called a *ground predicate*. For instance, with an assignment for a predicate $a_t = (c_1, c_2)$, we can simply write a ground predicate $t(c_1, c_2) = t(a_t)$. Similarly, a formula with an assignment to its arguments is called a *ground formula* (e.g., $f(a_f)$). Based on the probabilistic logic representations, MLN can thus be defined formally as a joint distribution over all possible assignments in set $\mathcal{A}_f$ for the formula set $\mathcal{F}$:

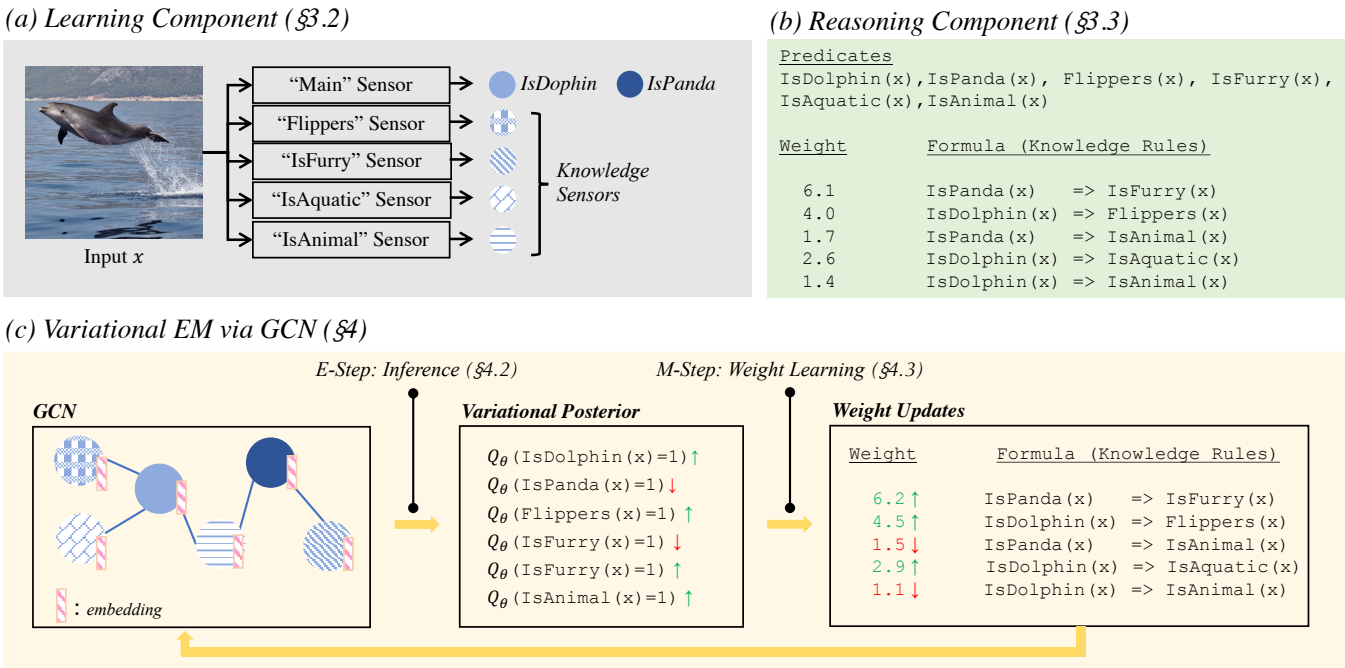


Fig. 1. The overview of our *learning with reasoning* framework CARE.

$$P_w(t_1, ..., t_L) = \frac{1}{Z(w)} \exp\left(\sum_{f \in \mathcal{F}} w_f \sum_{a_f \in \mathcal{A}_f} \phi_f(a_f)\right), \quad (1)$$

where $t_1, ..., t_L$ denote the L ground predicates (with assignment $\mathcal{A}_f$) that are used to form the formulas, w_f represents the corresponding weight for each formula, Note that $t_i(x)$ is a predicate function given input x, and for notation simplicity we will use t_i throughout this work to represent $t_i(x)$ when there is no ambiguity. $\phi_f(\cdot)$ represents the potential function for the given assignment, which takes 1 when the formula is true and 0 when it is false, and $Z(w)$ is the *partition function* summing over all possible assignments. Based on the grounding predicates, we can define a *possible world* by assigning a truth value to each possible ground predicate.

Robustness certification. The robustness certification technique aims to provide a certified robustness guarantee: given a robust radius $r \in \mathbb{R}_+$, any perturbation within r will not change the classifier's prediction [39], [40]. Formally, such technique takes a classifier $h : \mathbb{R}^d \to \mathcal{Y}$ and a clean (i.e., unperturbed) input x. It outputs r such that $h(x) = h(x')$ holds for any perturbed input x' with $d(x, x') < r$ under a specific metric d (e.g., ℓ_p norm). We provide more details about existing robustness certification techniques in our related work Section VI. In this paper, we mainly utilize the randomized smoothing technique [13], one of the state-of-the-art certification methods that can scale to large-scale datasets [36], [39], [41], to evaluate the certified robustness for different learning pipelines as follows. First, we will wrap a given learning model h to a new smoothed model $g(x) = \arg\max_{c \in \mathcal{Y}} \mathbb{P}(h(x + \delta) = c)$ where the $\delta \sim \mathcal{N}\left(0, \sigma^2 I\right)$ and the σ here control the variance of the added noise. Then, the resulting Gaussian smoothed classifier $g(x)$ can be certified by leveraging Neyman-Pearson lemma with no further assumption. Assume p_A is the probability of the returning class c_A, i.e., $p_A = \mathbb{P}\left(h(x + \delta) = c_A\right)$, and p_B is the "runner-up" probability, i.e., $p_B = \max_{c \neq c_A} \mathbb{P}(h(x+\varepsilon) = c)$, the smoothed classifier g is robust around x with the radius [13]:

$$R = \frac{\sigma}{2}\left(\Phi^{-1}\left(p_A\right) - \Phi^{-1}\left(p_B\right)\right), \quad (2)$$

where Φ^{-1} is the inverse of the standard Gaussian CDF. That is to say, it is guaranteed that there is no further adversarial perturbation within R, and thus the robustness can be certified.

III. CARE: Scalable Robust Learning with Reasoning

In this section, we first provide an overview of the proposed *learning with reasoning* pipeline CARE, followed by the detailed construction of the *learning* and *reasoning* components within the pipeline.

A. Overview of CARE

To effectively integrate domain knowledge into statistical machine learning models (e.g., DNNs), we propose CARE, which consists of a *learning* and a *reasoning* components. In particular, the *learning* component consists of a *main sensor* which serves for the main classification task and makes a multi-class prediction given an input; and several *knowledge sensors* which make predictions for the individual semantic objects requested by different knowledge rule given the same input. For instance, if we want to integrate the knowledge "Panda is furry" into the learning process, we will train a main sensor to predict the class of the input (e.g., different animal categories), and a knowledge sensor to predict whether the input "is furry", respectively. We then represent the knowledge as the first-order logic rule "`IsPanda(x)` $\implies$ `IsFurry(x)`" via a reasoning component.

Concretely, the *reasoning* component can be realized by different probabilistic graphical models, such as Markov logic

networks (MLN) and Bayesian networks. In this work, we focus on MLN as the reasoning component for different applications. However, it is known that the inference of MLN is computationally expensive due to the exponential cost of constructing the ground Markov network. Thus, we propose a scalable variational inference approach based on GCN to approximate the inference of MLN (Section IV). In particular, we propose an EM algorithm to jointly improve the accuracy of GCN in terms of learning the MLN (E step) and optimize the weights of formulas in the latent MLN towards better inference-time robustness (M step). Moreover, we consider different types of domain knowledge, such as attributes-based knowledge and category hierarchy knowledge, to improve the robustness of CARE (details in Section V).

Since the proposed CARE can be viewed as a general machine learning pipeline, we are able to certify its robustness using standard certification approaches [13], [35], [36], [38]. As the pure data-driven based machine learning approaches have reached a bottleneck for certified robustness so far due to the lack of additional information or prior knowledge, here we show that the proposed CARE is able to *significantly* improve the certified robustness on datasets including the high-resolution dataset AwA2 [30], the standard Word50 [31], the GTSRB [32] for road sign classification, and the security application of PDF Malware classification [33]. In addition, we will also show that simply adding more prediction models as an ensemble without explicit knowledge integration or reasoning, which is shown to obtain only marginal robustness improvement by existing work [11], [37], will not help achieve such high performance on certified robustness. We believe such knowledge integration and enabling reasoning ability is a promising way to break the existing robustness barriers.

B. Learning in CARE

Within the *learning* component of CARE, we construct a set of statistical learning models (e.g., DNNs, logistic regression, SVMs, etc.) to predict the main classification task and other knowledge sensors' classification tasks. For instance, as shown in Fig 1, the goal is to perform the animal category classification. In particular, we will train one main sensor to predict the main animal classes, say, *IsPanda*, *Dolphin*, and others. In order to integrate domain knowledge and logical reasoning ability into this learning process, we need to embed domain knowledge, such as "`IsPanda(x)` $\implies$ `IsFurry(x)`", into the pipeline. Thus, we will train a knowledge sensor to predict "whether the input is furry". Similarly, we will train other knowledge sensors for different knowledge rules. Here the main/knowledge sensors can be viewed as predicates, and the output of each knowledge sensor is a binary truth value; while for multiclass classification, we will map the d-way prediction of the main sensor to several binary truth values (detailed mapping process and constraints in Section IV).

Formally, we define the prediction output of i-*th* sensor $t_i(\cdot)$ as t_i, and the corresponding prediction confidence as z_i. Given an input x, the corresponding sensor predictions $t_i(x)$ are shown in Figure 1 (i.e., $t_i(x) = \texttt{IsFurry}(x)$).

C. Reasoning in CARE

Given an input x and predictions from different sensors $t_i(x)$, we will connect these predictions based on their logical relationships to enable the *reasoning* ability of our learning pipeline CARE. Such a logical relationship can be realized by different types of probabilistic graphical models, and in this paper, we will focus on MLN.

Concretely, as mentioned above, we will construct one main sensor and several knowledge sensors $t_i(x)$ as the predicates in MLN. We then build logical relationships among the predicates to form different formulas. Assume we have L sensors, an MLN will define a joint distribution based on the predefined logical formulas. For simplicity, we denote the collection of formulas as $\mathcal{F}$, and thus the joint distribution defined by MLN can be represented as below simplified from Equation (1):

$$P_w(t_1, ..., t_L) := \frac{1}{Z(w)} \exp\left(\sum_{f \in \mathcal{F}} w_f f(t_1, ..., t_L)\right), \quad (3)$$

where $Z(w)$ denotes the partition function summing over all the possible assignments of the predicates. Since in our learning pipeline, each formula will only have one unique correct assignment by construction to ensure robustness, we can simplify this joint distribution based on Equation (1).

The reasoning component of CARE can handle logic formulas expressed as first-order logic rules. In this paper, we further optimize for four types of logic rules that popularly used in practice as follows.

- *Attribute rule* ($t_i \implies t_j \vee t_k \vee ...$): Some prediction classes have specific attributes, which can be leveraged to construct knowledge rules. For instance, one attribute rule could be `IsPanda(x)` $\implies$ `IsFurry(x)`.
- *Hierarchy rule* ($t_i \implies t_j$): In general, there exist hierarchical relationships between different classes, based on which we can build formula $f(t_i, t_j) = \neg t_i \vee t_j$. For instance, `IsDog(x)` $\implies$ `IsAnimal(x)`, or a slightly more complicated example as `IsChihuahua(x)` $\vee$ `IsCollie(x)` $\vee$ `IsDalmatian(x)` $\implies$ `IsDog(x)`. For instance, we can build such hierarchical knowledge rules based on relationships extracted from WordNet.
- *Exclusion rule* ($t_i \oplus t_j$): Some class predictions are naturally exclusive from others. For instance, an animal cannot be a panda and dolphin at the same time, so we will exclude the possible world where `IsPanda`(x) and `IsDolphin`(x) are both true. In particular, we will introduce constraint `IsPanda`$(x) \wedge$ `IsDolphin`$(x) =$ `False`.

Next, we will discuss the *weight training* of sensors and formulas. For the weight of sensors, we aim to take the influence of the prediction confidence z_i of sensor $t_i(\cdot)$ into account, and thus we assign $\log[z_i/(1 - z_i)]$ to be the weight of sensor t_i. As a result, if there is no other formula, the marginal probability of the predicate t_i to be true will be its corresponding prediction confidence z_i. For other formulas, we will train the weights given observed variables based on variational EM steps (M-step) based on a trained GCN, and the details are illustrated in Section IV-C.

IV. Scalable Reasoning via Variational Inference Using GCN

In this section, we will illustrate in detail how to approximate the inference of our reasoning component MLN via variational inference based on GCN.

A. Variational EM Based on GCN

In order to conduct efficient inference and learning for MLN, existing work has introduced different approaches, including variational inference and Monte Carlo sampling [42], [43]. In particular, as MLN models the joint probability distribution of all predicates as defined in Eq. 3, it is possible to train the weights of knowledge rules (formulas) w within MLN by maximizing the log-likelihood of all the observed predicates (facts) $\log P_w(\mathcal{O})$. However, it is intractable to maximize the overall objective directly since it requires computing the whole partition function $Z(w)$ and integrating over all observed predicates $\mathcal{O}$ and unobserved ones $\mathcal{U}$. Thus, some works propose to instead optimize the variational evidence lower bound (ELBO) [29] of the data log-likelihood as below:

$$\log P_w(\mathcal{O}) \geq \mathcal{L}_{\text{ELBO}}(Q_\theta, P_w) := \mathbb{E}_{Q_\theta(\mathcal{U}|\mathcal{O})}\left[\log P_w(\mathcal{O},\mathcal{U})\right] - \mathbb{E}_{Q_\theta(\mathcal{U}|\mathcal{O})}\left[\log Q_\theta(\mathcal{U}|\mathcal{O})\right], \tag{4}$$

where $Q_\theta(\mathcal{U}|\mathcal{O})$ represents the variational posterior distribution, and the equality in Eq 4 holds if $Q_\theta(\mathcal{U}|\mathcal{O})$ equals to the true posterior $P_w(\mathcal{U}|\mathcal{O})$. Here since the sensor output variables together with the knowledge rules among them can be represented as a knowledge graph, we will use graphical convolutional networks (GCN) to encode the posterior distribution $Q_\theta(\cdot)$.

Now we need to learn the weights of MLN w, based on which we will make the inference for MLN to enforce the reasoning process given the knowledge rules. Thus, we leverage a variational EM algorithm [44] to optimize the ELBO in Eq. 4. In particular, since the input x is unknown, all the variables such as $\texttt{IsPanda}(x)$ are unobserved. Thus, we consider all variables to be unobserved by setting $\mathcal{O}$ to be empty set $\emptyset$, and optimize over the outputs of sensors $\mathcal{T} = \{t_1, t_2, ..., t_L\}$ with the following optimization objective:

$$\mathcal{L}_{\text{ELBO}}\left(Q_\theta, P_w\right) := \mathbb{E}_{Q_\theta(\mathcal{T})}\left[\log P_w(\mathcal{T})\right] - \mathbb{E}_{Q_\theta(\mathcal{T})}\left[\log Q_\theta(\mathcal{T})\right], \tag{5}$$

which is the negative KL divergence between $Q_\theta(\mathcal{T})$ and $P_w(\mathcal{T})$. On the other hand, it can also be viewed as to directly approximate $P_w(\mathcal{T})$ via a variational distribution $Q_\theta(\mathcal{T})$.

Next, we will discuss in detail the EM steps. On the high level, in the E-step, we will fix the MLN weights w for the knowledge rules and optimize GCN parameters Q_θ to minimize the KL distance between $Q_\theta(\mathcal{T})$ and $P_w(\mathcal{T})$; in the M-step, we will fix Q_θ and update the weights w to maximize the the log-likelihood function $\mathbb{E}_{Q_\theta(\mathcal{T})}\left[\log P_w(\mathcal{T})\right]$. The E-step and the M-step will be executed alternately multiple times until convergence.

B. E-step: Optimizing Q_θ

In E-step, we aim to minimize the KL distance between the variational distribution $Q_\theta(\mathcal{T})$ and the true distribution $P_w(\mathcal{T})$. Since the inference of the MLN is #P-complete [20], we approximate $P_w(\mathcal{T})$ with a mean-field distribution, which has been shown to scale up to large graphical models [28], [29], [45]. In the mean-field variational distribution where the variables are independent, the joint distribution of outputs of sensors (unobserved variables) can be formed as the following,

$$Q_\theta(\mathcal{T}) := \prod\nolimits_{t_i \in \mathcal{T}} Q_\theta(t_i), \tag{6}$$

We constrain the sum of the $Q_\theta(t_i)$, whose associated class confidence z_i comes from the same main sensor (i.e., multi-class classifier), to be 1, in order to model key-constraints, namely the exclusion rules, induced by a d-way classifier.

To further improve the efficiency of inference and take into account the knowledge graph structure, we parameterize the Q_θ here with graph convolutional networks (GCNs), where θ represents the parameters of GCN. In particular, we will construct nodes for each class prediction based on both main and knowledge sensor outputs as shown in Figure 1 (c). In other words, each node will be associated with a scalar which represents the confidence of the corresponding class. However, the message passing between different nodes on the GCN will not be effective if the input feature is only a scalar, and thus it will be hard for the GCN to learn the variational posterior distribution well. So following [29], we also train a class embedding vector $\vec{\mu_i}$ for each node and use the scalar multiplication of the class prediction confidence z_i and the corresponding class embedding vector $\vec{\mu_i}$ as the input of each node for further encouraging the expressivity of the model.

Based on the mean-field approximation, and joint distribution $\log P_w(\mathcal{T}) = \log(\frac{1}{Z(w)}\exp(\sum_{f\in\mathcal{F}} w_f f(t_1, ..., t_L)))$, the ELBO from Eq. 5 can thus be rewritten as:

$$\mathcal{L}_{\text{ELBO}}\left(Q_\theta, P_w\right) = \mathbb{E}_{Q_\theta(\mathcal{P})}\left(\sum_{f\in\mathcal{F}} w_f f(t_1, ..., t_L) - \log Z(w)\right) - \mathbb{E}_{Q_\theta(\mathcal{T})} \log Q_\theta(\mathcal{T}). \tag{7}$$

Since the MLN weights w is fixed during the E-step, the $\log Z(w)$ here is a constant and can be ignored during the optimization. However, with this new optimization objective, we cannot obtain the gradient of it w.r.t. the parameters θ in GCN through backpropagation directly. Thus, we first derive the explicit form of the gradient as bellows, and the full proof is deferred to Appendix A1.

Lemma IV.1. *The gradient of $\mathcal{L}_{\text{ELBO}}\left(Q_\theta, P_w\right)$ w.r.t. the GCN parameters θ can be derived as:*

$$\mathbb{E}_{Q_\theta(\mathcal{T})}\left(\sum_{f\in\mathcal{F}} w_f f(t_1, ..., t_L) - \log Q_\theta(\mathcal{T})\right) \nabla_\theta \log Q_\theta(\mathcal{T}). \tag{8}$$

This shows that the gradient can be estimated through multiple sampling from $Q_\theta(\mathcal{T})$. In specific, the term $\nabla_\theta \log Q_\theta(\mathcal{T})$ can be directly derived through backpropagation, and then the question remains as how to calculate $\sum_{f\in\mathcal{F}} f(t_1, ..., t_L)$. For the single sensor (formula), it can be calculated by the dot multiplication of $\boldsymbol{t}$ and $\log[\boldsymbol{z}/(1-\boldsymbol{z})]$ where $\boldsymbol{t} = [t_1, ..., t_L]$; for the exclusion rule based formula, as mentioned before, we can imply it by constraining the sum of $Q_\theta(t_i)$ to be 1 for the corresponding classes. For the attribute rule and hierarchy

rule based formulas, they can be reduced to the combination of four basic kinds of formulas:

$$\begin{aligned} t_i &\implies t_j \vee t_k \vee ... \vee t_l, \\ t_i &\implies t_j \wedge t_k \wedge ... \wedge t_l, \\ t_j \vee t_k \vee ... \vee t_l &\implies t_i, \\ t_j \wedge t_k \wedge ... \wedge t_l &\implies t_i. \end{aligned} \tag{9}$$

We thus provide an efficient score calculation for them as follows, and the detailed proof is deferred to Appendix A2.

Theorem 1. *The function $\sum_{f\in\mathcal{F}} f(t_1, ..., t_L)$ based on the four types of formulas defined in Equation* (9) *can be efficiently calculated as follows:*

$$\sum_{f\in\mathcal{F}} w_f f(t_1, ..., t_L) = \boldsymbol{w}\,\mathrm{Neg}(A\boldsymbol{t}^T + B), \tag{10}$$

where $\boldsymbol{w}$ is the row vector of the concatenation of all w_f for $f \in \mathcal{F}$, $\mathrm{Neg}(\cdot)$ is an indicator function which maps the values larger than 0 to 0 and maps the other values to 1, A and B are the matrices determined by the pre-defined formulas with the shape $|\mathcal{F}| \times L$ and $|\mathcal{F}| \times 1$, respectively.

In practice, we will shift the term $\left(\sum_{f\in\mathcal{F}} w_f f(t_1, ..., t_L) - \log Q_\theta(\mathcal{T})\right)$ by subtracting the sample mean for reducing the variance in the estimation for the gradient with Monte Carlo based on the fact that $\mathbb{E}_{Q_\theta(\mathcal{T})} \nabla_\theta \log Q_\theta(\mathcal{T}) = 0$.

Since the tasks here are supervised, and there is label information for each input during training, we can add a supervised negative likelihood to encourage the overall learning and help guide the direction of the optimization:

$$\mathcal{L}_{\text{label}}\left(Q_\theta\right) = -\sum_{i=1}^{L} \log Q_\theta\left(\mathrm{GT}(t_i)\right), \tag{11}$$

where $\mathrm{GT}(t_i)$ is the corresponding ground truth for predicate t_i during training. Thus, the final E-step training optimization objective is:

$$\mathcal{L}_{\text{new}}(Q_\theta) = \mathcal{L}_{\text{ELBO}}\left(Q_\theta, P_w\right) - \eta \mathcal{L}_{\text{label}}\left(Q_\theta\right), \tag{12}$$

where η is a hyperparameter to balance the trade-off of these two likelihood terms. The embedding $\vec{\mu}$ will also be updated during the optimization of GCN through the chain rule for better expressiveness.

During the test stage, the prediction for each class will be based on the marginal probability $Q_\theta(t_i)$, and it can be seen as a knowledge-enhanced correction for the original prediction.

C. M-step: Optimizing w

In M-step, the GCN model is fixed, and we update the weight of the formula w by maximizing the log-likelihood function, i.e., $\mathbb{E}_{Q_\theta(\mathcal{T})}\left[\log P_w\left(\mathcal{T}\right)\right]$, which is to maximize the term

$$\mathbb{E}_{Q_\theta(\mathcal{T})} \log \frac{\exp\{\sum_{f\in\mathcal{F}} w_f f(t_1, ..., t_L)\}}{\sum_{t'_1,...t'_L} \exp\{\sum_{f\in\mathcal{F}} w_f f(t'_1, ..., t'_L)\}}. \tag{13}$$

However, the partition function, namely, the denominator that involves an integration of all the variables, is intractable to compute. We optimize the pseudo-likelihood [46] as an alternative, which is defined as:

$$P_w^*(t_1, ..., t_L) := \prod_{i=1}^{L} P_w\left(t_i | MB\left(t_i\right)\right), \tag{14}$$

Algorithm 1 The whole training procedure for the variational EM based on GCN.

Input: Input x, a set of trained sensors (predicates) $\mathcal{T}$, model GCN, sensor output confidence $\boldsymbol{z} = [z_1, ..., z_L]$, number of training epochs K

Output: Trained GCN; the weight of MLN formulas w

Initialize the node embedding of GCN $\boldsymbol{\mu} = [\vec{\mu_1}, ..., \vec{\mu_L}]$.
$\boldsymbol{m} = [\vec{m_1}, ..., \vec{m_L}] \leftarrow [z_1\vec{\mu_1}, ..., z_1\vec{\mu_L}]$ # Initialize node features in GCN.
$Q_\theta(\mathcal{T}) \leftarrow \mathrm{GCN}(\boldsymbol{m}; \theta)$ # Get variational distribution.
for $j = 1$ to K **do**
  $\theta \leftarrow \arg\max_\theta \mathcal{L}_{\text{new}}\left(Q_\theta, P_w\right)$ # E-step.
  Update node embedding $\boldsymbol{\mu}$ from $Q_\theta(\mathcal{T})$
  $\boldsymbol{m} = [\vec{m_1}, ..., \vec{m_L}] \leftarrow [z_1\vec{\mu_1}, ..., z_1\vec{\mu_L}]$ # Update the input feature to GCN.
  $Q_\theta(\mathcal{T}) \leftarrow \mathrm{GCN}(\boldsymbol{m}; \theta)$ # Update variational distribution.
  $w \leftarrow \arg\max_w \mathbb{E}_{Q_\theta}\left[\log P_w\left(\mathcal{T}\right)\right]$ # M-step.
end for
return GCN parameter θ; weight of MLN formulas w.

where $\mathrm{MB}(t_i)$ is the Markov blanket of the predicate t_i. In other words, $\mathrm{MB}(t_i)$ is the set of formulas where the predicate t_i appears. Then, following [20], given a formula f, the gradient of the pseudo-log-likelihood w.r.t. its associated weight w, namely $\frac{\partial}{\partial w} \log P_w^*(t_1, ..., t_L)$, is

$$\begin{aligned} \sum_{i=1}^{L} \left[f(t_1, ..., t_L) - P_w\left(t_i = 0 | MB\left(t_i\right)\right) f\left([t_i = 0]\right)\right. \\ \left. - P_w\left(t_i = 1 | MB\left(t_i\right)\right) f\left([t_i = 1]\right)\right], \end{aligned} \tag{15}$$

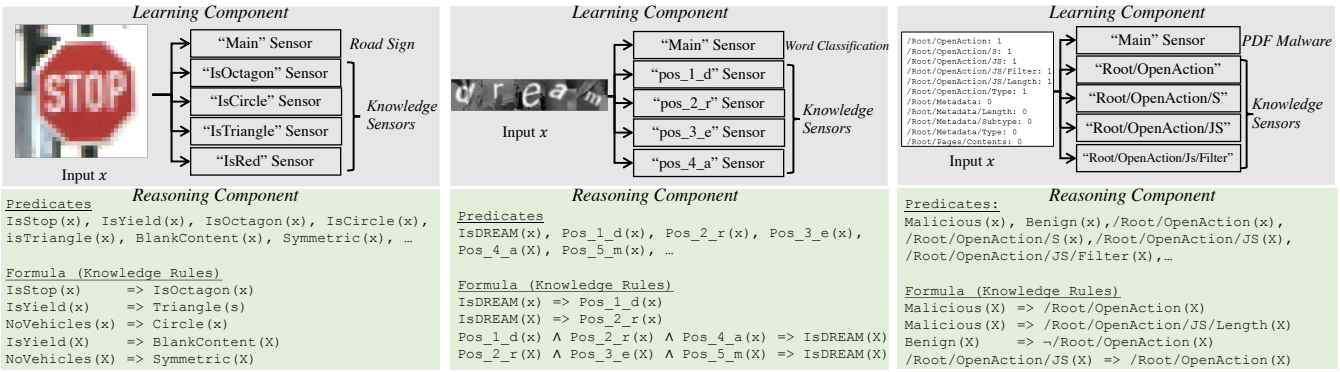
where $f\left([t_i = 0]\right)$ represents the truth value of the formula f when we force $t_i = 0$ while leaving the remaining $t_{j,j\neq i}$ unchanged; similar for $f\left([t_i = 1]\right)$. Finally, we will maximize the original intractable log-likelihood function through optimizing the expectation of the pseudo-log-likelihood $\mathbb{E}_{Q_\theta(\mathcal{T})}\left[\log P_w^*\left(\mathcal{T}\right)\right]$, and the gradient w.r.t. the weight w of the formula will be estimated through multiple sampling from the variational distribution Q_θ. The algorithm for the whole training pipeline is provided in Algorithm 1.

V. Experimental Evaluation

In this section, we present experimental evaluation of CARE on four large datasets: Animals with Attributes (AwA2) [30], Word50 [31], GTSRB [32], and PDF malware dataset from Contagio [33]. The illustration of these datasets and the corresponding construction of CARE are shown in Figures 1 to 4. With the knowledge integration and reasoning, CARE achieves significantly higher certified robustness than the state-of-the-art methods under different radii. We also conduct a set of ablation studies to explore the influence of the number of integrated knowledge rules, the empirical robustness of CARE, and the explanation properties of CARE via case studies. All experiments are run on four GeForce RTX 2080 Ti GPUs.

A. Experimental Setup

Datasets and the implementation of *learning* component. For AwA2, all sensors, including the main sensor and knowledge sensors, are trained with the architecture of ResNet-50 [47]; while for Word50 and PDF malware datasets, we



(a) GTSRB Classification **(b) Word50 Classification** **(c) PDF Malware Classification**

Fig. 2. Learning and reasoning components of CARE on GTSRB, Word50, and PDF malware classification. (AwA2 is illustrated in Figure 1).

TABLE I

CERTIFIED ACCURACY UNDER DIFFERENT ℓ_2 PERTURBATION RADII ON AWA2 DATASET.

σ	Method	ACR	Certified Accuracy under Radius r												
			0.00	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00	2.20	2.40
0.25	Gaussian	0.544	84.0	77.6	71.4	58.6	40.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SWEEN	0.552	84.2	78.8	71.2	60.8	43.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SmoothAdv	0.574	78.6	74.8	71.6	69.4	62.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Consistency	0.587	81.6	78.2	74.0	69.8	58.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	MultiTask	0.593	79.8	78.2	76.2	71.0	58.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE	**0.709**	**96.6**	**94.2**	**91.4**	**85.4**	**75.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.50	Gaussian	0.827	75.6	71.2	64.6	58.2	53.0	46.2	38.8	32.0	21.2	0.0	0.0	0.0	0.0
	SWEEN	0.854	76.4	73.8	67.8	60.4	53.6	47.4	39.6	34.6	22.4	0.0	0.0	0.0	0.0
	SmoothAdv	0.949	72.0	69.8	66.6	62.8	60.2	56.8	52.2	47.6	40.2	0.0	0.0	0.0	0.0
	Consistency	0.953	74.0	71.2	68.8	64.6	61.2	56.0	51.2	46.8	40.4	0.0	0.0	0.0	0.0
	MultiTask	0.842	69.6	67.6	63.2	58.2	53.4	49.4	42.4	36.8	27.2	0.0	0.0	0.0	0.0
	CARE	**1.141**	**91.2**	**88.2**	**84.2**	**78.8**	**73.4**	**68.4**	**63.2**	**56.2**	**44.0**	0.0	0.0	0.0	0.0
1.00	Gaussian	0.994	59.6	54.6	51.6	49.0	44.8	40.8	36.6	32.6	29.6	26.4	22.8	20.0	17.2
	SWEEN	1.059	62.2	57.6	54.8	50.2	45.8	41.8	39.2	34.4	32.0	29.0	26.8	22.0	18.8
	SmoothAdv	1.231	57.2	54.0	53.0	49.8	47.2	45.4	42.2	40.8	38.2	36.8	34.0	32.6	30.2
	Consistency	1.247	54.0	52.0	50.0	48.0	45.6	44.0	42.0	40.6	39.4	37.8	36.0	33.8	31.6
	MultiTask	1.192	51.6	49.8	48.4	46.8	46.0	45.0	42.0	40.0	38.2	36.0	34.0	31.2	29.2
	CARE	**2.127**	**87.0**	**85.2**	**84.0**	**82.0**	**80.4**	**78.2**	**75.6**	**71.4**	**68.6**	**65.8**	**61.8**	**59.4**	**56.0**

build the feed-forward neural network with two hidden layers activated by ReLU for all sensors. Specifically, for Word50, following the similar setting in [31], the number of hidden neurons is set to 512 for character classification and 1024 for word classification; for the PDF malware dataset, following the same setting in [48], the number of hidden neurons is set to 200 for both main sensor and the knowledge sensors. For GTSRB, we use the GTSRB-CNN [6] for all sensors.

The implementation of *reasoning* component. The dimension of the embedding $\vec{\mu}$ for each predicate is fixed to 512, and it is initialized with the He uniform initialization [49]. For all datasets, we use the GCN with two hidden layers, and the hidden dimension is also set to 512. For the construction of the graph, we introduce a node for each predicate, and each predicate corresponds to one class that appeared in the main sensor or knowledge sensor exactly. The edge will connect associated predicates that appeared in the same knowledge rule (formula). We train the GCN with 60 epochs, and the learning rate is set to 0.01 in the first 40 epochs and set to 0.001 for the last 20 epochs.

Baselines. For the dataset AwA2, Word50, and GTSRB, whose features are continuous, we consider four state-of-the-art ℓ_2 certification baselines: (1) *Gaussian smoothing* [13] trains smoothed classifiers by directly augmenting the input images with the Gaussian noise during training; (2) *SWEEN* [37] employs weighted ensemble to improve the certified robustness; (3) *SmoothAdv* [35] further applies adversarial training to improve the certified robustness based on the Gaussian smoothing; (4) *Consistency* [36] adds a consistency regularization term in the training loss based on standard Gaussian smooth training; (5) *MultiTask* incorporates the additional sensor information by adding more classification heads in the main sensor and training it together with the knowledge classification tasks under Gaussian augmentation. For the PDF malware dataset, whose features are binary, we consider the state-of-the-art ℓ_0 certification baseline: Lee et al. [38]. To train a ℓ_0 smoothed robust model, we smooth each feature by replacing it with a range of discrete random values with probability $(1-\alpha)$ following existing work [38]. SWEEN serves as another baseline to represent the SOTA robust ensembles.

Evaluation metrics. We mainly focus on the *certified robustness* of different methods. For AwA2, Word50, and GTSRB, we report the certified accuracy of CARE and other pipelines under different ℓ_2 radius r, following the standard certification setting [13]. For the PDF malware dataset, we will report the certified accuracy under different ℓ_0 radius r, which follows the certification setting in [38]. In addition, we also

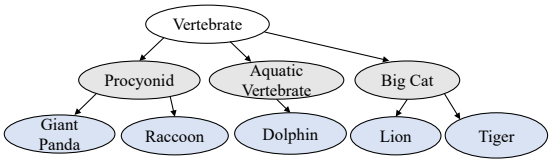


Fig. 3. Hierarchy knowledge rule of AwA2. The blue nodes represent the main task of animal categories, and the grey nodes represent part of the knowledge sensors.

report the average certified radius (ACR) following [50]. The whole certification procedure is provided in Appendix B2.

B. Evaluation on AwA2

Dataset description. AwA2 [30] contains 37322 images of 50 animal categories and provides 85 class attributes for each class. For example, animal *"Fox"* is assigned with the attribute labels such as *"brown"*, *"has a tail"* and *"has no spots"*. In addition to the attribute knowledge, we also construct hierarchical relations between the classes based on WordNet [34] as another type of domain knowledge.

Task and the implementation of *learning* component. The main task here is to classify the 50 animal classes for the input image. First, we will train one main sensor for classifying these 50 animals and train 85 knowledge sensors for classifying each binary attribute, respectively. We utilize WordNet [34] to build a hierarchy tree by iteratively searching the inherited hypernyms of the 50 leaf animal classes, part of the nodes are shown in Figure 3. Then, we perform additional hierarchy classification tasks on the 28 internal nodes (gray nodes in Figure 3). We conduct additional ablation studies to evaluate the effect of training different numbers of knowledge sensors in Section V-F. The final sensing vector $\boldsymbol{z}$ is of $50 + 85 + 28 = 163$ dimensions by concatenating the output confidence of all the sensors (predicates).

The implementation of *reasoning* component. For each possible sensor task, we introduce a corresponding predicate. Thus, the number of the predicates is 163 here.

For the *attribute-based* knowledge/formulas, we let each class imply its owned attributes. For example, if the animal class *"raccoon"* has the attributes *gray* and *furry*, we will construct two formulas: $\texttt{IsRaccoon}(x) \implies \texttt{IsGray}(x)$ and $\texttt{IsRaccoon}(x) \implies \texttt{IsFurry}(x)$. The total number of attribute-based formulas is the number of possible attributes of all animals, which is 1562.

For the *hierarchy-based* knowledge/formulas, the classes in the internal nodes (the gray node as shown in Figure 3) will imply at least one of its children to be true. For example, $\texttt{IsProcyonid}(x) \implies \texttt{IsPanda}(x) \vee \texttt{IsRaccoon}(x)$; $\texttt{IsBigCat}(x) \implies \texttt{IsLion}(x) \vee \texttt{IsTiger}(x)$. The number of hierarchy-based formulas is the number of the internal nodes, which is 28, and thus the total number of the overall formulas is $1562 + 28 = 1590$.

Certification details. All the images are resized to 224×224 for training the sensors. We randomly sample 80% images from each animal class as the training data while picking 10 images from each class within the remaining unsampled set for certification. Following the standard setting [13], we certify these 500 images with confidence 99.9% (the results are certified with $N = 10,000$ samples of smoothing noise). We test all methods based on three levels of smoothing noise $\sigma = 0.25, 0.50, 1.00$. For small $\sigma = 0.25$, the η in Equation (12) is set to 0.2, and for $\sigma = 0.50$ and 1.00, the η is set to 0.6. Generally, with larger training noise, the η needs to be larger to help maintain the benign accuracy and guide the training of GCN. The details for other baselines are deferred to Appendix B1.

Certification results. The certification results of CARE and baselines are shown in Table I, and as we can see, our method CARE improves the certified accuracy under different radii with different smoothing levels. In addition, we can also replace the main sensor of CARE with different training methods, and detailed results are in Appendix C. Out of interest, we also explore the importance of the reasoning module by comparing it with the method that replaces the GCN in CARE with a simple linear classifier while maintaining others as the same, the corresponding results are provided in Appendix D.

C. Evaluation on Word50

Dataset description. Word50 [31] is created by randomly choosing 50 words, and each consists of 5 lower case characters, which are extracted from the Chars74K dataset [51]. The background of each character is inserted with random patches, and the whole character image is further perturbed with scaling, rotation, and translation to increase the difficulty of recognition, making it more challenging than traditional digit recognition tasks. The size of each character image is 28×28, and examples of the word images are shown in Figure 2 (b). The intriguing property of this dataset is that the relationship between nearby characters can be treated as prior knowledge to help build reliable predictions.

Task and the implementation of *learning* component. We conduct two tasks here, one is for the *word* classification, and one is for the *character* classification. We train one main sensor for classifying the 50 words and 5 knowledge sensors for classifying the character on each position of the word. The sensing vector $\boldsymbol{z}$ can be represented as $[\boldsymbol{u}; \boldsymbol{e}_1; ...; \boldsymbol{e}_5]$, where $\boldsymbol{u} \in \{0,1\}^{50}$ is the output confidence of the main sensor with 50 dimensions, $\boldsymbol{e}_i \in \{0,1\}^{26}$ is the output confidence of the knowledge sensor which classifies the character on the i-th position with 26 dimensions. Therefore, the total dimension of the sensing vector $\boldsymbol{z}$ here is $50 + 26 \times 5 = 180$.

The implementation of *reasoning* component. For each word prediction, we will use a predicate to represent it. For instance, we will use $\texttt{DREAM}(x)$ to denote if the input word is DREAM. While for the character appeared in the word, we will use $\texttt{Pos_i_\$}(x)$ to represent if the character appeared in the i-th position of the input image x is character '\$'. For example, the predicate $\texttt{Pos_2_a}(x)$ predicts if the second character appeared in the input word image is 'a'. Thus, the number of the predicates is 180.

Given the 50 known words, we will construct knowledge rules based on the word and the corresponding characters in each position. For the attribute-based knowledge/formulas, we will build them like $\texttt{IsDREAM}(x) \implies \texttt{Pos_1_d}(x)$, ..., $\texttt{DREAM}(x) \implies \texttt{Pos_5_m}(x)$, and the total number of such

TABLE II

CERTIFIED ACCURACY UNDER DIFFERENT ℓ_2 PERTURBATION RADII ON WORD CLASSIFICATION IN WORD50.

σ	Method	ACR	Certified Accuracy under Radius r												
			0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.10	1.20
0.12	Gaussian	0.115	48.6	38.0	26.4	16.6	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SWEEN	0.152	51.4	44.6	36.6	27.4	18.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SmoothAdv	0.197	59.0	53.8	45.2	38.0	29.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Consistency	0.157	53.4	45.2	36.0	28.4	20.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	MultiTask	0.142	55.6	43.0	32.6	23.6	15.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE	**0.391**	**97.0**	**96.0**	**91.4**	**81.4**	**70.4**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.25	Gaussian	0.125	42.0	32.6	24.0	17.6	13.0	8.0	4.6	2.8	2.2	0.6	0.0	0.0	0.0
	SWEEN	0.194	48.2	41.8	35.6	28.4	22.0	16.6	10.8	7.6	4.8	2.4	0.0	0.0	0.0
	SmoothAdv	0.246	55.6	47.0	40.0	35.8	29.0	24.8	17.6	11.8	8.4	4.8	0.0	0.0	0.0
	Consistency	0.201	47.6	40.2	36.0	29.0	24.0	17.4	13.2	9.0	6.6	2.4	0.0	0.0	0.0
	MultiTask	0.166	49.2	40.8	34.2	25.6	17.4	11.8	7.2	4.2	2.0	1.2	0.0	0.0	0.0
	CARE	**0.674**	**97.2**	**94.8**	**92.6**	**89.4**	**81.8**	**73.6**	**64.4**	**55.2**	**43.6**	**30.8**	0.0	0.0	0.0
0.50	Gaussian	0.082	27.8	20.4	14.6	11.2	7.8	4.4	3.4	2.2	1.6	1.0	0.4	0.4	0.4
	SWEEN	0.143	35.0	29.0	23.8	17.8	14.2	11.0	8.6	6.4	4.4	3.8	2.0	1.4	0.8
	SmoothAdv	0.168	38.0	32.6	27.0	21.0	16.0	12.6	9.4	7.4	6.0	5.4	3.8	2.8	1.6
	Consistency	0.146	34.8	28.4	22.8	17.8	13.0	9.8	8.8	7.8	5.8	4.6	3.2	2.4	1.6
	MultiTask	0.108	32.0	24.8	18.6	13.4	9.8	8.2	5.2	4.2	2.8	1.8	1.2	0.4	0.4
	CARE	**0.697**	**87.6**	**84.4**	**78.4**	**73.6**	**69.0**	**63.0**	**56.6**	**50.0**	**44.0**	**36.6**	**30.0**	**24.0**	**18.0**

TABLE III

CERTIFIED ACCURACY UNDER DIFFERENT ℓ_2 PERTURBATION RADII ON CHARACTER CLASSIFICATION IN WORD50.

σ	Method	ACR	Certified Accuracy under Radius r												
			0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.10	1.20
0.12	Gaussian	0.234	72.0	64.0	53.8	45.8	34.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SWEEN	0.250	72.8	64.2	58.6	50.2	39.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SmoothAdv	0.228	63.8	59.0	52.0	45.4	37.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Consistency	0.226	66.2	59.0	51.6	44.4	37.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	MultiTask	0.191	62.8	53.6	45.2	35.4	25.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE	**0.341**	**90.2**	**85.2**	**78.0**	**70.8**	**60.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.25	Gaussian	0.290	63.4	57.2	51.0	42.2	35.0	26.8	20.2	13.4	9.0	4.2	0.0	0.0	0.0
	SWEEN	0.315	65.8	58.6	52.4	46.0	37.8	30.0	23.6	16.6	10.8	7.2	0.0	0.0	0.0
	SmoothAdv	0.289	55.8	50.0	44.6	37.6	33.4	27.4	23.6	19.8	15.0	10.0	0.0	0.0	0.0
	Consistency	0.285	57.0	52.2	46.2	39.6	35.2	27.8	23.8	16.6	11.0	5.6	0.0	0.0	0.0
	MultiTask	0.246	60.2	53.2	44.6	35.8	28.8	21.8	15.2	8.8	5.4	2.4	0.0	0.0	0.0
	CARE	**0.539**	**87.6**	**83.2**	**77.4**	**73.0**	**63.2**	**55.6**	**50.2**	**41.8**	**32.4**	**21.2**	0.0	0.0	0.0
0.50	Gaussian	0.165	40.0	33.6	29.8	23.6	18.2	13.2	9.2	7.2	4.8	2.8	1.8	0.6	0.6
	SWEEN	0.184	42.0	36.8	31.4	25.2	20.4	14.8	11.4	8.4	5.2	3.6	2.8	1.2	0.8
	SmoothAdv	0.165	31.2	27.0	23.6	20.8	16.8	14.6	11.6	9.8	7.4	5.0	4.6	2.8	1.2
	Consistency	0.162	37.0	32.4	27.0	21.2	17.0	14.2	10.0	6.8	4.8	3.0	2.2	1.0	0.8
	MultiTask	0.209	47.6	42.2	35.0	27.4	23.0	17.0	13.2	9.6	6.4	4.4	3.0	1.6	1.6
	CARE	**0.539**	**80.6**	**76.8**	**70.4**	**65.4**	**59.2**	**53.0**	**44.8**	**37.0**	**29.6**	**21.6**	**17.2**	**11.6**	**6.6**

formulas is $50 \times 5 = 250$. In addition, for this dataset, we find that the identification of the characters on at least three positions is enough to determine the whole word, thus we also construct the knowledge/formulas like $\texttt{Pos_1_d}(x) \wedge \texttt{Pos_4_a}(x) \wedge \texttt{Pos_5_m}(x) \implies \texttt{IsDREAM}(x)$ for further enriching the prediction robustness. The number of such formulas is $50 \times (\binom{5}{3} + \binom{5}{4} + \binom{5}{5}) = 800$, and thus the number of the overall formulas is $250 + 800 = 1050$.

Certification results. The certified accuracy on word-level and character-level classifications are shown in Table II and Table III, respectively. As we can see, CARE significantly outperforms all other baselines under different perturbation radii and smoothing noise levels. Using different models for the main sensor in CARE can be found in Appendix C. The training and the certification details are deferred to Appendix B1 and Appendix B3, respectively.

D. Evaluation on GTSRB

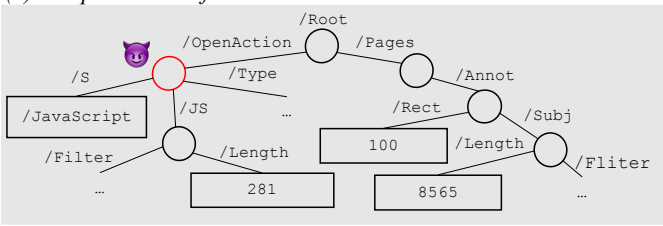
Dataset description. Here we use the GTSRB dataset [32] following [21], which contains 12 types of the road signs: *"Stop"*, *"Priority Road"*, *"Yield"*, *"Construction Area"*, *"Keep Right"*, *"Turn Left"*, *"Do not Enter"*, *"No Vehicles"*, *"Speed Limit 20"*, *"Speed Limit 50"*, *"Speed Limit 120"*, *"End of Previous Limitation"*. Each image is resized to 32×32 for training, and an example is shown in Figure 2 (a).

Task and the implementation of *learning* component. The main task is to classify the 12 types of German road signs. For the learning component, first, we will train a main sensor to classify those 12 road signs. Next, we manually construct 20 knowledge sensors such as `IsOctagon()`, `IsSqaure()`, and `IsCircle()` based on the border patterns and the contents of the road signs. The full knowledge construction is provided in Appendix B4. The final sensing vector $\boldsymbol{z}$ is of $12 + 20 = 32$ dimensions by concatenating the output confidence of all the sensors.

The implementation of *reasoning* component. For each road sign and attribute (e.g., `IsStop()` and `IsOctagon()`), we will build one associated predicate and thus the number of predicates is 32.

For the attribute-based knowledge/formulas, following [21], we treat the attribute that only one of the road signs

(a) The parse tree of a PDF malware



(b) Learning Component

```
Features: /Root/OpenAction/S: 1, /Root/OpenAction/JS: 1,
/Root/OpenAction/Type: 1, /Root/OpenAction/JS/Filter: 1,…
Predict /Root/OpenAction: 0(attacked) => 1
```

(c) Reasoning Component

```
Predicates:
Malicious(X), Benign(X),
/Root/OpenAction(X),/Root/OpenAction/S(X),/Root/OpenAction/JS(X),
/Root/OpenAction/JS/Filter(X),/Root/OpenAction/JS/Length(X),
/Root/OpenAction/Type(X),/Root/Pages,/Root/Pages/Annot(X), …

Formulas (Knowledge Rules)
Malicious(X) => /Root/OpenAction(X)
Malicious(X) => /Root/OpenAction/S(X)
Malicious(X) => /Root/OpenAction/JS/Length(X)
Benign(X) => ¬/Root/OpenAction(X)
Benign(X) => ¬/Root/OpenAction/S(X)
Benign(X) => ¬/Root/OpenAction/JS(X)
/Root/OpenAction/JS(X) => /Root/OpenAction(X)
/Root/OpenAction/JS/Filter(X) => /Root/OpenAction/JS(X)
/Root/Pages(X) => /Root/Pages/Annot(X), …
```

Fig. 4. (a) The parsed tree structure of a PDF malware and an attack example; (b) the learning component of CARE under attack; (c) the reasoning component of CARE and some examples of corresponding knowledge rules.

TABLE IV
CERTIFIED ACCURACY ON GTSRB UNDER ℓ_2 RADII.

Method	Certified Accuracy under ℓ_2 Radius r												
	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.10	1.20
Gaussian	97.9	96.5	92.6	86.8	82.5	78.6	74.3	68.7	63.2	57	53.3	50.4	47.7
SWEEN	99.2	97.1	94.7	87.9	82.9	78.8	74.1	69.1	65.8	58.6	55.8	52.5	49.6
SmoothAdv	97.1	95.9	91.8	86.6	82.9	78.6	**75.3**	71.0	66.9	**63.0**	56.8	53.9	51.6
Consistency	99.4	98.8	95.3	90.7	83.3	78.0	74.3	71.0	65.6	60.5	57.4	54.5	51.2
MultiTask	96.5	94.4	90.3	85.8	81.3	76.3	71.8	67.3	62.8	57.8	55.3	51.4	48.6
CARE	**99.6**	**99.2**	**96.7**	**91.2**	**84.4**	**79.4**	**75.3**	**72.0**	**67.9**	**63.0**	**57.8**	**55.1**	**51.9**

owns as the permissive attribute and allow each of them to imply its associated road sign. For example, only stop sign is of octagon shape, and thus the corresponding formulas would be $\texttt{IsOctagon}(x) \implies \texttt{IsStop}(x)$. Next, we treat the remaining attributes as preventative attributes and let each road sign imply them. For example, both *"do not enter"* sign and *"no vehicles"* sign are circle, so the corresponding formulas are constructed like $\texttt{IsDoNotEnter}(x) \implies \texttt{IsCircle}(x)$ and $\texttt{IsNoVehicles}(x) \implies \texttt{IsCircle}(x)$. The detailed knowledge construction is in Appendix B4, and the final number of the constructed formulas here is 44.

The hierarchy knowledge/formulas are constructed between the attributes that have inclusion relations. For examples, both the attributes octagon and square are one kind of polygon instead of circle, so we will construct two formulas $\texttt{IsOctagon}(x) \implies \texttt{IsPolygon}(x)$ and $\texttt{IsSquare}(x) \implies \texttt{IsPolygon}(x)$. The full inclusion relations between the attributes are provided in Appendix B4, and the final number of the hierarchy formulas is 14, so the total number of formulas is $44 + 14 = 58$.

Certification results. We report the best certified accuracy of each method under $\sigma \in \{0.12, 0.25, 0.50\}$ for each radius, and the results are shown in Table IV. As we can see, CARE is consistently better than if not equal to the best of the baseline approaches under different perturbation radii. The full certification results for each method under different σ are in Appendix C. The training and the certification details are deferred to Appendix B1 and Appendix B3, respectively.

E. Evaluation on PDF Malware Dataset

Dataset description. The PDF malware dataset from Contagio [33] contains 16800 clean and 11960 malicious PDFs. Following the standard setting in [48], we use Hidost [52] to extract the binary structural path features from the parsed

TABLE V
CERTIFIED ACCURACY ON PDF MALWARE UNDER ℓ_0 RADII.

Method	Certified Accuracy under Radius r									
	0	1	2	3	4	5	6	7	8	9
Lee et al. [38]	**99.8**	99.0	96.1	80.0	80.0	68.0	46.5	15.1	5.7	5.7
SWEEN	**99.8**	99.0	**97.7**	85.2	80.3	72.5	57.2	22.6	8.9	8.9
MultiTask	99.7	99.0	97.2	82.8	80.5	72.7	59.0	53.8	9.9	9.9
CARE	99.5	**99.3**	96.9	**85.5**	**84.2**	**77.4**	**63.4**	**54.5**	**13.5**	**13.5**

tree structure of each PDF with the default `compact` path option [53]. An example of the parsed tree structure and the corresponding extracted binary Hidost features are shown in Figure 4 (a) and Figure 4 (b) respectively. The final extracted features have 3514 dimensions, and the feature in each position is a binary value indicating the existence of a specific path.

Task and the implementation of *learning* component. The main task is to detect PDF malware. First, we train one main sensor based on the whole 3514 binary features extracted from PDF with Hidost. Then, we manually pick 6 malicious features shared with most malicious PDFs but not in most benign PDFs and 8 benign features shared with most benign PDFs but not in most malicious PDFs. Detailed information on these 14 selected features is provided in Appendix B5. We assume the adversary can arbitrarily manipulate some of the whole 3514 features. The final sensing vector $\boldsymbol{z}$ is the concatenation of the output confidence of all the sensor output confidence with $2 + 14 = 16$ dimensions.

The implementation of *reasoning* component. As shown in Figure 4 (c), first, we construct the predicates $\texttt{IsMalicious}(x)$ and $\texttt{IsBenign}(x)$ to indicate if the input PDF x is malicious or benign. Then, for each of the 14 picked features we will construct one predicate to indicate if the structural path exists, such as $\texttt{/Root/OpenAction}(x)$, $\texttt{/Root/OpenAction/JS}(x)$, $\texttt{/Root/Metadata}(x)$. And thus, the number of predicates is 16 here.

For the 6 malicious features, we let the malicious PDF imply each of them, while the benign PDF will imply the non-existence of them. For instance, $\texttt{IsMalicious}(x) \implies \texttt{/Root/OpenAction/JS}(x)$ and $\texttt{IsBenign}(x) \implies \neg\texttt{/Root/OpenAction/JS}(x)$. Similarly for the 8 benign features: $\texttt{IsBenign}(x) \implies \texttt{/Root/Metadata/JS}(x)$ and $\texttt{Malicious}(x) \implies \neg\texttt{/Root/Metadata/JS}(x)$. The total number of the attribute-based formulas here is $14 \times 2 = 28$, and more details in Appendix B5

TABLE VI
CERTIFIED ACCURACY OF CARE USING DIFFERENT NUMBER OF KNOWLEDGE SENSORS ON AWA2.

σ	Method	ACR	Certified Accuracy under Radius r												
			0.00	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00	2.20	2.40
0.25	Gaussian	0.544	84.0	77.6	71.4	58.6	40.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE-10	0.594	89.4	84.6	76.2	65.8	46.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE-30	0.639	93.8	89.2	82.4	71.6	52.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE-50	0.671	94.8	91.6	87.8	77.6	55.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE-70	0.703	96.0	**94.2**	90.2	83.4	65.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE-All	**0.709**	**96.6**	**94.2**	**91.4**	**84.8**	**67.4**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.50	Gaussian	0.827	75.6	71.2	64.6	58.2	53.0	46.2	38.8	32.0	21.2	0.0	0.0	0.0	0.0
	CARE-10	0.892	79.2	75.0	69.2	62.4	56.2	50.0	43.6	36.0	25.6	0.0	0.0	0.0	0.0
	CARE-30	0.956	85.0	80.0	75.0	65.8	60.6	53.6	47.8	37.6	28.2	0.0	0.0	0.0	0.0
	CARE-50	1.015	88.0	82.8	78.0	72.2	65.6	58.2	51.0	41.4	29.0	0.0	0.0	0.0	0.0
	CARE-70	1.047	89.0	86.4	80.4	73.2	67.4	60.4	50.6	43.0	31.6	0.0	0.0	0.0	0.0
	CARE-All	**1.114**	**91.2**	**88.2**	**84.2**	**78.8**	**71.2**	**66.4**	**56.8**	**46.8**	**34.6**	0.0	0.0	0.0	0.0
1.00	Gaussian	0.994	59.6	54.6	51.6	49.0	44.8	40.8	36.6	32.6	29.6	26.4	22.8	20.0	17.2
	CARE-10	1.196	67.6	62.0	59.2	55.2	50.4	46.6	44.6	41.6	37.8	33.8	29.2	26.0	21.8
	CARE-30	1.542	76.8	73.8	71.6	67.8	64.8	61.0	57.0	51.6	48.4	45.2	40.8	38.0	32.4
	CARE-50	1.796	81.8	80.2	76.8	74.6	72.8	69.0	67.4	62.6	57.6	54.0	49.0	45.4	41.6
	CARE-70	2.033	86.0	**85.2**	82.6	80.8	78.6	76.4	72.4	70.4	67.4	64.2	57.8	53.8	50.2
	CARE-All	**2.092**	**87.0**	**85.2**	**84.0**	**82.0**	**80.4**	**78.2**	**75.6**	**71.2**	**68.0**	**64.4**	**61.0**	**57.0**	**52.8**

TABLE VII
EMPIRICAL ROBUST ACCURACY OF DIFFERENT METHODS ON AWA2, WORD50, AND GTSRB UNDER ℓ_2 ATTACKS.

Method	AwA2				Word50				GTSRB			
	σ	ϵ			σ	ϵ			σ	ϵ		
		1.8	2.4	3.0		0.4	0.8	1.2		0.6	1.2	1.8
Gaussian	0.25	29.8	16.6	8.6	0.12	10.4	1.0	0.0	0.12	67.9	41.8	31.3
SWEEN		33.0	17.4	10.4		27.8	11.2	3.8		78.0	58.2	46.1
SmoothAdv		48.4	36.4	25.4		30.0	8.4	2.2		72.2	51.9	39.9
Consistency		40.4	29.0	17.8		21.0	4.6	0.6		73.3	52.9	52.3
CARE		**66.4**	**42.4**	**26.2**		**86.4**	**82.4**	**80.4**		**79.8**	**58.8**	**52.5**
Gaussian	0.50	40.4	31.4	21.4	0.25	13.6	2.2	0.0	0.25	72.0	47.7	28.2
SWEEN		43.0	33.4	23.0		27.6	14.4	3.2		72.6	49.4	32.1
SmoothAdv		46.0	38.6	31.6		29.8	10.2	2.4		73.9	54.1	35.4
Consistency		47.0	37.0	29.4		24.2	8.2	0.4		73.7	51.0	34.8
CARE		**67.2**	**55.6**	**44.6**		**90.0**	**84.2**	**80.2**		**75.1**	**56.2**	**37.4**
Gaussian	1.00	39.2	32.8	25.8	0.50	12.6	3.6	1.0	0.50	67.3	45.9	26.5
SWEEN		40.0	34.2	28.6		20.0	10.6	4.4		67.7	47.5	29.4
SmoothAdv		39.4	34.0	30.8		20.0	9.8	3.6		65.6	49.4	34.6
Consistency		39.8	35.0	31.6		15.8	7.6	3.2		70.2	50.8	32.5
CARE		**76.8**	**73.4**	**68.6**		**80.0**	**72.6**	**63.2**		**72.0**	**56.6**	**35.0**

In addition, we also construct the hierarchy-based formulas based on the parsed PDF tree structure, which constructs 12 more formulas. In specific, for each internal node in the parsed tree, every descendant of it will imply its existence. For instance, `/Root/OpenAction/JS`(x) $\implies$ `/Root/OpenAction`(x), `/Root/Metadata/Length`(x) $\implies$ `/Root/Metadata`(x). The number of the overall formulas is $28 + 12 = 40$.

Certification results. We report the best certified accuracy of each method under $\alpha \in \{0.80, 0.85, 0.90\}$ for each radius where α is defined in Section V-A, and the results are shown in Table V. As we can see, with knowledge integration, CARE enhances the certified robustness significantly, especially under large ℓ_0 perturbation. The full certification results for each method under different α are in Appendix C. The training and the certification details are deferred to Appendix B1 and Appendix B3, respectively.

F. Ablation Studies

In addition to the improved certified robustness of CARE, we conduct a set of ablation studies to further explore the impact of the integrated number of knowledge rules, the empirical robustness of different methods, and the interesting explanation properties of CARE.

Number of knowledge rules. Generally, the certified robustness will improve with the increase of the number of integrated knowledge rules. Here we quantitatively analyze this phenomenon on the AwA2 dataset. In particular, we randomly pick k knowledge sensors and build the formulas based on their corresponding provided knowledge rules. Then, we will retrain the GCN with these limited confidence vectors and knowledge rules. For simplicity, we denote CARE-k as the model with k knowledge sensors. As shown in Table VI, the certified robustness of CARE improves with the increase of the number of knowledge rules (sensors). Interestingly, CARE still beats the baselines with only 10 knowledge sensors. However, simply increasing the number of the base model for the ensemble method will only provide marginal performance as shown in Appendix E, and this phenomenon has also been theoretically proven in [11].

Empirical robustness. Except for the certified robustness, we also evaluate the empirical robustness of CARE on AwA2, Word50, and GTSRB to demonstrate the effectiveness of our method further. Concretely, we will first attack the main sensor to generate adversarial instances and then test them on the CARE pipeline. Since the main sensor here is a smoothed model, we follow [54] to sample 100 Monte Carlo samples to estimate the gradient, and the detailed procedure of the empirical attack is provided in Appendix F.

We provide the experimental results for both ℓ_2 and ℓ_∞ attacks on our model and report the empirical robust accuracy on the main task (i.e., Animal class prediction on AwA; word prediction on Word50; road sign prediction on GTSRB) for baselines and our method. Under ℓ_2 attack, the number of update steps in PGD is fixed to 100, and the attack step size is set to 0.2. The ℓ_2 attack results under different perturbation magnitude ϵ for dataset AwA2, Word50 and GTSRB are shown in Table VII. The results for ℓ_∞ attack and the exploration for attacking with the attributes are deferred to Appendix F.

Transferability between sensors. The large improvement of the empirical robustness with knowledge integration can be attributed to: (a) low attack transferability between different sensors; (b) high difficulty of attacking all sensors at the same time such that their predictions still satisfy the knowledge rules among them. Here we evaluate the attack transferability between 12 sensors (one main sensor and eleven random picked attribute sensors, all are trained under $\sigma = 0.50$) on AwA2 under ℓ_2 perturbation size $\epsilon = 3.0$, and the corresponding results are shown in Figure 6 (in Appendix F). All the sensors here are trained under Gaussian noise with $\sigma = 0.5$, and the result in the i-th row and the j-th column represents the empirical robust accuracy when we conduct a PGD attack on the i-th sensor while testing the obtained adversarial images on j-th sensor. As we can see, when we attack one of the sensors, only a few other sensors will be influenced, and the reason can be attributed to the fact that different knowledge sensors will rely on different features. As the adversary usually can only flip a small set of the sensors, which could be again corrected by the knowledge rules, the overall pipeline CARE is still robust against perturbations.

Case studies on AwA2. To better demonstrate the knowledge correction process in CARE, we provide some case studies against adversarial attacks. As shown in Figure 5, the ground label for the input is *"Dolphin"* and it is predicted correctly by the main sensor originally (Figure 5 (b)); then we perform a strong targeted attack with 100 update steps

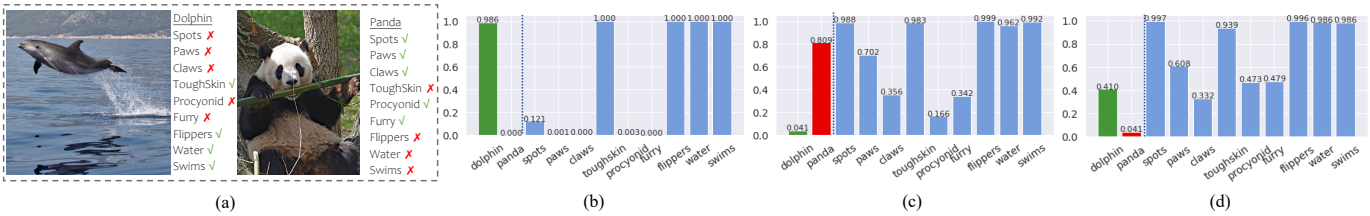


Fig. 5. A case study on AwA2. (a) Attributes for dolphin and panda classes; (b) prediction confidence of a DNN before attack; (c) prediction confidence of the DNN after adversarial attack; (d) prediction confidence of CARE given the same adversarial input. (The ground truth is *"dolphin"*).

to make it be misclassified as *"Panda"* (Figure 5 (c)). All the sensors here are trained under $\sigma = 0.25$, the perturbation size ϵ and the attack step size here are set to 6.0 and 0.4, respectively. Some attributes for these two classes are shown in Figure 5 (a). As we can see, although some attributes like *"Spots"* and *"Paws"* are attacked, most of the attributes remain unattacked (e.g. *"Toughskin"*, *"Flippers"*, *"Water"*, *"Swims"*), and thus after passing through the CARE pipeline, the knowledge rules among these attributes would correct the sensor predictions (Figure 5 (d)). More examples are shown in Appendix G.

VI. Related Work

Knowledge integration and logical reasoning. There is abundant domain knowledge in real-world data. For instance, the labels in ImageNet [55] contains a semantic hierarchy structure based on the lexical database WordNet [34]. Thus, how to quantitatively represent and effectively integrate such knowledge is an important research direction. In particular, Bayesian logic programs [56], relational Markov networks [57] and Markov logic networks [20] have been used for knowledge reasoning. In addition, with the development of deep learning, some works have started to introduce structured logic rules into a neural network to improve performance. For example, Deng et al. [58] construct hierarchy and exclusion graphs, which are based on the hierarchical relations between classes, to improve the classification on ImageNet. Hu et al. [59] develop a distillation method to encode the knowledge into the weight of neural networks. However, leveraging such domain knowledge and relationships to improve the certified robustness of DNNs has not been well explored yet, and this work provides the first learning with a reasoning pipeline to improve the *certified* robustness of DNNs.

Markov Logic Networks. MLNs, which extend the probabilistic graphical model with first-order logic, has been largely used for solving the problems like collective classification [60], link prediction [61] and entity resolution [62]. However, the inference of MLN is #P-complete, and it can be either solved with variable elimination-based methods like belief propagation [63], [64] and junction tree algorithm [65] or approximated by random sampling like Markov chain Monte Carlo (MCMC) [24] and importance sampling [66]. Nevertheless, MLN is still hard to be scaled for large knowledge graphs in practice, and the combination of deep neural networks and MLN is still constrained on small dataset [22]. Therefore, our method CARE aims to provide a more robust and scalable framework for such MLN-based logical reasoning via variational inference [27] and equip it with a more powerful posterior parameterization by graph neural network.

Graph Neural Networks. GNN [67] is well recognized for its superior performance in handling large-scale knowledge graphs and the effective encoding ability. Different from classic knowledge graph embedding such as TransE [68], DistMult [69], and RotatE [70], which cannot leverage prior domain knowledge, GNN models such as Graph Convolutional Network (GCN) [23] can be used to learn semantically-constrained embeddings [71]. In addition, Qu et al. [28] propose a Graph Markov Neural Network (GMNN), which combines GNN with a conditional random field to improve the performance of semi-supervised object classification and link classification in relational data. These works have provided valuable experience in projecting traditional graphical models to deep neural networks for efficient training and inference.

Certified robustness. The robustness certification aims to ensure that the prediction of a classifier is consistent within a certain perturbation radius [39]. Currently, there are mainly two types of certification methods. The *complete* certification, which guarantees to find the perturbation if it exists, is usually based on satisfiability modulo theories [72], [73], or mixed integer-linear programming [74], [75]. However, the exact certification is `NP-complete` for feed-forward networks. The *incomplete* certification, guarantees to find non-certifiable instances, while may miss some certifiable ones based on different relaxed optimization. With such relaxation, incomplete certification is usually more practical and efficient, which is mainly based on linear programming relaxation [18], [76] or semi-definite programming [77], [78]. However, these incomplete certification approaches are only applicable for specific architecture and can not scale to a large dataset like ImageNet. Later, Cohen et al. [13] provide a *probabilistic* certification method based on randomized smoothing, which can be scaled to ImageNet and is further improved with adversarial training [35] and consistency regularization [36].

VII. Conclusion

In this work, we propose the first scalable certifiably robust machine learning pipeline CARE by integrating knowledge to enable reasoning ability for reliable prediction. We show that when combining learning with reasoning, CARE can effectively scale to large datasets and achieve both high certified robustness and empirical robustness. We believe our observations and findings will inspire interesting future directions on leveraging domain knowledge to improve ML robustness.

REFERENCES

[1] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[3] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *AAAI*, 2018.

[4] C. Xiao, J. Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," in *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[5] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. Chen, M. Liu, and B. Li, "Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks," in *2021 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2021, pp. 1302–1320. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP40001.2021.00076

[6] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

[7] B. J. Erickson, P. Korfiatis, Z. Akkus, and T. L. Kline, "Machine learning for medical imaging," *Radiographics*, vol. 37, no. 2, p. 505, 2017.

[8] G. D. Magoulas and A. Prentza, "Machine learning in medical applications," in *Advanced course on artificial intelligence*. Springer, 1999, pp. 300–307.

[9] C. Xiao, R. Deng, B. Li, F. Yu, M. Liu, and D. Song, "Characterizing adversarial examples based on spatial consistency information for semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 217–234.

[10] Z. Yang, L. Li, X. Xu, S. Zuo, Q. Chen, P. Zhou, B. I. P. Rubinstein, C. Zhang, and B. Li, "Trs: Transferability reduced ensemble via promoting gradient diversity and model smoothness," in *Advances in Neural Information Processing Systems*, 2021.

[11] Z. Yang, L. Li, X. Xu, B. Kailkhura, T. Xie, and B. Li, "On the certified robustness for ensemble models and beyond," in *International Conference on Learning Representations*, 2022.

[12] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[13] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1310–1320.

[14] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 3–14.

[15] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International conference on machine learning*. PMLR, 2018, pp. 274–283.

[16] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv preprint arXiv:1810.12715*, 2018.

[17] H. Zhang, H. Chen, C. Xiao, S. Gowal, R. Stanforth, B. Li, D. Boning, and C.-J. Hsieh, "Towards stable and efficient training of verifiably robust neural networks," in *International Conference on Learning Representations*, 2019.

[18] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang, "A convex relaxation barrier to tight robustness verification of neural networks," *Advances in Neural Information Processing Systems*, vol. 32, pp. 9835–9846, 2019.

[19] G. Yang, T. Duan, J. E. Hu, H. Salman, I. Razenshteyn, and J. Li, "Randomized smoothing of all shapes and sizes," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 693–10 705.

[20] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1, pp. 107–136, 2006.

[21] N. M. Gürel, X. Qi, L. Rimanic, C. Zhang, and B. Li, "Knowledge enhanced machine learning pipeline against diverse adversarial attacks," *ICML*, 2021.

[22] Z. Yang, Z. Zhao, H. Pei, B. Wang, B. Karlas, J. Liu, H. Guo, B. Li, and C. Zhang, "End-to-end robustness for sensing-reasoning machine learning pipelines," *arXiv preprint arXiv:2003.00120*, 2020.

[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[24] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. CRC press, 1995.

[25] F. Niu, C. Zhang, C. Ré, and J. Shavlik, "Scaling inference for markov logic via dual decomposition," in *2012 IEEE 12th International Conference on Data Mining*, 2012, pp. 1032–1037.

[26] K. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," *arXiv preprint arXiv:1301.6725*, 2013.

[27] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[28] M. Qu, Y. Bengio, and J. Tang, "Gmnn: Graph markov neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 5241–5250.

[29] Y. Zhang, X. Chen, Y. Yang, A. Ramamurthy, B. Li, Y. Qi, and L. Song, "Efficient probabilistic logic reasoning with graph neural networks," in *International Conference on Learning Representations*, 2019.

[30] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2251–2265, 2018.

[31] L.-C. Chen, A. Schwing, A. Yuille, and R. Urtasun, "Learning deep structured models," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1785–1794.

[32] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.

[33] M. Parkour, "16,800 clean and 11,960 malicious files for signature testing and research." [Online]. Available: http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html

[34] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[35] H. Salman, J. Li, I. P. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang, "Provably robust deep learning via adversarially trained smoothed classifiers," in *NeurIPS*, 2019.

[36] J. Jeong and J. Shin, "Consistency regularization for certified robustness of smoothed classifiers," in *34th Conference on Neural Information Processing Systems (NeurIPS) 2020*. NeurIPS committee, 2020.

[37] C. Liu, Y. Feng, R. Wang, and B. Dong, "Enhancing certified robustness via smoothed weighted ensembling," *arXiv preprint arXiv:2005.09363*, 2020.

[38] G.-H. Lee, Y. Yuan, S. Chang, and T. Jaakkola, "Tight certificates of adversarial robustness for randomly smoothed classifiers," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[39] L. Li, T. Xie, and B. Li, "Sok: Certified robustness for deep neural networks," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, 22-26 May 2023*. IEEE, 2023.

[40] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, M. J. Kochenderfer *et al.*, "Algorithms for verifying deep neural networks," *Foundations and Trends® in Optimization*, vol. 4, no. 3-4, pp. 244–404, 2021.

[41] N. Carlini, F. Tramer, J. Z. Kolter *et al.*, "(certified!!) adversarial robustness for free!" *arXiv preprint arXiv:2206.10550*, 2022.

[42] P. Domingos and D. Lowd, "Unifying logical and statistical ai with markov logic," *Communications of the ACM*, vol. 62, no. 7, pp. 74–83, 2019.

[43] Z. Liu and G. von Wichert, "A generalizable knowledge framework for semantic indoor mapping based on markov logic networks and data driven mcmc," *Future Generation Computer Systems*, vol. 36, pp. 42–56, 2014.

[44] Z. Ghahramani and M. J. Beal, "Graphical models and variational methods," in *Advanced Mean Field Methods-Theory and Practice*. Citeseer, 2000.

[45] M. Qu and J. Tang, "Probabilistic logic neural networks for reasoning," *Advances in neural information processing systems*, vol. 32, 2019.

[46] J. Besag, "Statistical analysis of non-lattice data," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 24, no. 3, pp. 179–195, 1975.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[48] Y. Chen, S. Wang, D. She, and S. Jana, "On training robust {PDF} malware classifiers," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2343–2360.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[50] R. Zhai, C. Dan, D. He, H. Zhang, B. Gong, P. Ravikumar, C.-J. Hsieh, and L. Wang, "Macer: Attack-free and scalable robust training via maximizing certified radius," in *International Conference on Learning Representations*, 2019.

[51] T. E. De Campos, B. R. Babu, M. Varma *et al.*, "Character recognition in natural images." *VISAPP (2)*, vol. 7, no. 2, 2009.

[52] N. Šrndic and P. Laskov, "Detection of malicious pdf files based on hierarchical document structure," in *Proceedings of the 20th Annual Network & Distributed System Security Symposium*. Citeseer, 2013, pp. 1–16.

[53] "Hidost: Toolset for extracting document structures from pdf and swf files," https://github.com/srndic/hidost.

[54] J. Hayes, "Extensions and limitations of randomized smoothing for robustness guarantees," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 786–787.

[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[56] K. Kersting and L. D. Raedt, "Towards combining inductive logic programming with bayesian networks," in *International Conference on Inductive Logic Programming*. Springer, 2001, pp. 118–131.

[57] B. Taskar, P. Abbeel, M.-F. Wong, and D. Koller, "Relational markov networks," *Introduction to statistical relational learning*, vol. 175, p. 200, 2007.

[58] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam, "Large-scale object classification using label relation graphs," in *European conference on computer vision*. Springer, 2014, pp. 48–64.

[59] Z. Hu, X. Ma, Z. Liu, E. H. Hovy, and E. P. Xing, "Harnessing deep neural networks with logic rules," in *ACL (1)*, 2016.

[60] R. Crane and L. McDowell, "Investigating markov logic networks for collective classification." in *ICAART (1)*, 2012, pp. 5–15.

[61] P. Domingos and M. Richardson, "1 markov logic: A unifying framework for statistical relational learning," *Statistical Relational Learning*, p. 339, 2007.

[62] P. Singla and P. Domingos, "Entity resolution with markov logic," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 572–582.

[63] J. S. Yedidia, W. Freeman, and Y. Weiss, "Generalized belief propagation," *Advances in neural information processing systems*, vol. 13, 2000.

[64] J. S. Yedidia, W. T. Freeman, Y. Weiss *et al.*, "Understanding belief propagation and its generalizations," *Exploring artificial intelligence in the new millennium*, vol. 8, no. 236-239, pp. 0018–9448, 2003.

[65] D. Kahle, T. Savitsky, S. Schnelle, and V. Cevher, "Junction tree algorithm," *Stat*, vol. 631, 2008.

[66] D. Venugopal and V. G. Gogate, "Scaling-up importance sampling for markov logic networks," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[67] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[68] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[69] R. Kadlec, O. Bajgar, and J. Kleindienst, "Knowledge base completion: Baselines strike back," *ACL 2017*, p. 69, 2017.

[70] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *International Conference on Learning Representations*, 2018.

[71] Y. Xie, Z. Xu, M. S. Kankanhalli, K. S. Meel, and H. Soh, "Embedding symbolic knowledge into deep networks," *Advances in neural information processing systems*, vol. 32, 2019.

[72] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 97–117.

[73] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 269–286.

[74] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *arXiv preprint arXiv:1706.07351*, 2017.

[75] M. Fischetti and J. Jo, "Deep neural networks as 0-1 mixed integer linear programs: A feasibility study," *arXiv preprint arXiv:1712.06174*, 2017.

[76] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *NeurIPS*, 2018.

[77] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations*, 2018.

[78] A. Raghunathan, J. Steinhardt, and P. S. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," in *NeurIPS*, 2018.

[79] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

APPENDIX

A. Proof Details

1) Proof of Lemma IV.1:

Proof.

$$\nabla_\theta \mathbb{E}_{Q_\theta(\mathcal{T})}\left(\sum_{f\in\mathcal{F}} w_f f(t_1,...,t_L) - \log Z(w) - \log Q_\theta(\mathcal{T})\right)$$
$$= \nabla_\theta \int \left(Q_\theta(\mathcal{T})\left(\sum_{f\in\mathcal{F}} w_f f(t_1,...,t_L)\right) - Q_\theta(\mathcal{T})\log Q_\theta(\mathcal{T})\right) d\boldsymbol{t}$$
$$= \int Q_\theta(\mathcal{T})\nabla_\theta \log Q_\theta(\mathcal{T})\left(\sum_{f\in\mathcal{F}} w_f f(t_1,...,t_L)\right)$$
$$- Q_\theta(\mathcal{T})\log Q_\theta(\mathcal{T})\nabla_\theta \log Q_\theta(\mathcal{T}) - Q_\theta(\mathcal{T})\nabla_\theta \log Q_\theta(\mathcal{T})\, d\boldsymbol{t}$$
$$= \mathbb{E}_{Q_\theta(\mathcal{T})}\left(\sum_{f\in\mathcal{F}} w_f f(t_1,...,t_L) - \log Q_\theta(\mathcal{T}) - 1\right)\nabla_\theta \log Q_\theta(\mathcal{T})$$

Further, with the truth $\mathbb{E}_{Q_\theta(\mathcal{T})}\nabla_\theta \log Q_\theta(\mathcal{T}) = 0$, we can see the term $\left(\sum_{f\in\mathcal{F}} w_f f(t_1,...,t_L) - \log Q_\theta(\mathcal{T}) - 1\right)$ shown above can be shifted by any constant without changing the whole expectation, which just means we can ignore the -1 inside this term. □

2) Proof of Theorem 1:

Proof. We only need to prove that for each type of the formula defined in Equation (9), the truth value of it can be written in the form of $\mathrm{Neg}(\boldsymbol{a}\boldsymbol{t}^T + b)$ where $\boldsymbol{a}$ is a row vector with shape $1 \times n$ and b is a constant.

Then, for Type 1 formula, the truth value of it can be directly calculated by $\mathrm{Neg}\left(t_i - (t_j + t_k + ... + t_l)\right)$; and for Type 2 formula, the truth value can be calculated by $\mathrm{Neg}\left(t_i - (t_j + t_k + ... + t_l)/m\right)$ where m is the number of the appeared $t_j, t_k, ..., t_l$ here; for Type 3 formula, the truth value can be calculated by $\mathrm{Neg}\left(-t_i + (t_j + t_k + ... + t_l)/m\right)$; for Type 4 formula, the truth value can be calculated by $\mathrm{Neg}\left(-t_i + (t_j + t_k + ... + t_l) - m + 1\right)$.

the proof still holds for the cases with negation on some predicates like $\neg t_i$, which is equivalent to replacing the t_i above with $1 - t_i$. □

B. Experiment Details

1) Training details: For AwA2, the sensor is initialized with the weight pretrained on ImageNet and finetuned with a learning rate 0.001 for 30 epochs; the batch size is also set to 256. For Word50, the sensor is trained with 90 epochs, and the initial learning rate is set to 0.01 and will be decayed by 0.1 at 30-th and 60-th epoch, and the batch size is set to 128. For GTSRB, the model is trained with 150 epochs, and the initial learning rate is set to 0.01 and will be decayed by 0.1 at 50-th and 100-th epoch, the batch size is set to 200. For the PDF malware dataset, the sensor is trained with 90 epochs, and the initial learning rate is set to 0.05 and will be decayed by 0.1 at 30-th and 60-th epoch, and the batch size is set to 128. for all image datasets, we balance the number of training images from each class during the training. For MultiTask, we add more classification heads in the main sensor and train it together with other knowledge tasks under Gaussian noise; the loss is defined as the mean of the classification loss for each task, the training epoch is set to 150, the initial learning rate is still set the same for each dataset and will be decayed by 0.1 at 50-th and 100-th epoch.

Algorithm 2 SAMPLEUNDERNOISE(f, x, n, σ).

Input: Base classifier f, clean input image x, the number of smoothing noise n, smoothing noise magnitude σ.
Output: A vector of class counts.

```
counts ← [0, 0, ..., 0]
for i = 1 to n do
    x_rs ← x + N(0, σ²I)
    y ← f(x_rs)
    counts[y] += 1
end for
return counts
```

Algorithm 3 Certification Procedure for Randomized Smoothing.

Input: The magnitude of the smoothing noise σ, the magnitude of the local smoothing noise σ', the base classifier f, the number of the smoothing noise for selection n_0, the number of the smoothing noise for estimation n, the certification confidence $(1-\alpha)$.
Output: Certified prediction and its robust radius.

```
counts0 ← SAMPLEUNDERNOISE(f, x, n0, σ)
ĉ_A ← top index in counts0
counts ← SAMPLEUNDERNOISE(f, x, n, σ)
p_A ← LOWERCONFBOUND(counts[ĉ_A], n, 1 − α)
if p_A > 1/2 then
    return ĉ_A and radius σ Φ⁻¹(p_A)
else
    return ABSTAIN
end if
```

The number of the base models in SWEEN is fixed to 6 for all experiments. And for the training with SmoothAdv, the ϵ is set to 255 and the m is set to 2 under all sigmas on the datasets AwA2 and Word50; while on GTSRB, the ϵ is set to 127 under $\sigma = 0.12, 0.25$ and is set to 255 under $\sigma = 0.50$. For the training with Consistency, the λ is set to 10 and the m is set to 2 on AwA2 under all sigmas; while on Word50, the m is set to 2 under all sigmas, and the λ is set to 10 for $\sigma = 0.12, 0.25$ and is set to 5 for $\sigma = 0.50$; for GTSRB, the m and the λ are set to 2 and 5, respectively under all sigmas.

2) Certification Procedure: The whole certification process is provided in Algorithm 3 following [13], the auxiliary function SAMPLEUNDERNOISE is shown in Algorithm 2 and the LOWERCONFBOUND$(k, n, 1-\alpha)$ is a function which returns a one-sided $(1-\alpha)$ lower confidence bound $\underline{p}$ for the Binomial parameter p given $k \sim \mathrm{Binomial}(n, p)$.

3) Certification details: **Word50 certification details:** The training, validation, and test sets contain $10,000$, $2,000$ and

TABLE VIII
CERTIFIED ACCURACY FOR CARE UNDER DIFFERENT ℓ_2 PERTURBATION RADII ON THE AWA2 DATASET.

σ	Method	ACR	Certified Accuracy under Radius r												
			0.00	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00	2.20	2.40
0.25	CARE (Gaussian)	**0.709**	**96.6**	**94.2**	**91.4**	84.8	67.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE (SmoothAdv)	0.707	95.4	92.4	89.8	**85.4**	**75.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE (Consistency)	0.693	95.0	92.0	87.2	83.0	70.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.50	CARE (Gaussian)	1.114	**91.2**	**88.2**	**84.2**	**78.8**	71.2	66.4	56.8	46.8	34.6	0.0	0.0	0.0	0.0
	CARE (SmoothAdv)	**1.141**	88.2	85.2	80.8	**78.8**	**73.4**	67.6	62.2	54.8	43.2	0.0	0.0	0.0	0.0
	CARE (Consistency)	1.138	87.8	84.6	80.0	76.8	**73.4**	**68.4**	**63.2**	**56.2**	**44.0**	0.0	0.0	0.0	0.0
1.00	CARE (Gaussian)	2.092	**87.0**	**85.2**	**84.0**	**82.0**	**80.4**	**78.2**	**75.6**	71.2	68.0	64.4	61.0	57.0	52.8
	CARE (SmoothAdv)	2.087	85.0	83.0	81.6	79.6	76.6	75.0	73.2	**71.4**	68.0	64.8	59.2	56.0	53.8
	CARE (Consistency)	**2.127**	85.4	84.0	83.0	80.2	78.4	76.2	73.4	70.6	**68.6**	**65.8**	**61.8**	**59.4**	**56.0**

TABLE IX
CERTIFIED WORD ACCURACY FOR CARE UNDER DIFFERENT ℓ_2 PERTURBATION RADII ON WORD50.

σ	Method	ACR	Certified Accuracy under Radius r												
			0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.10	1.20
0.12	CARE (Gaussian)	0.360	94.8	89.6	82.6	75.8	62.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE (SmoothAdv)	0.371	93.2	89.4	85.0	79.4	67.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE (Consistency)	**0.391**	**97.0**	**96.0**	**91.4**	**81.4**	**70.4**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.25	CARE (Gaussian)	0.624	96.0	92.4	88.0	82.2	75.6	68.8	58.0	48.0	37.8	25.6	0.0	0.0	0.0
	CARE (SmoothAdv)	0.577	91.6	86.4	81.0	74.6	69.2	63.0	53.6	45.4	35.6	24.0	0.0	0.0	0.0
	CARE (Consistency)	**0.674**	**97.2**	**94.8**	**92.6**	**89.4**	**81.8**	**73.6**	**64.4**	**55.2**	**43.6**	**30.8**	0.0	0.0	0.0
0.50	CARE (Gaussian)	0.671	87.0	83.0	77.2	73.4	68.2	60.6	54.2	48.2	40.6	34.0	28.0	20.6	15.0
	CARE (SmoothAdv)	0.690	85.2	82.0	77.4	71.6	66.6	60.8	54.2	48.0	42.6	**36.6**	29.4	**24.0**	**18.0**
	CARE (Consistency)	**0.697**	**87.6**	**84.4**	**78.4**	**73.6**	**69.0**	**63.0**	**56.6**	**50.0**	**44.0**	36.4	**30.0**	21.8	16.2

TABLE X
CERTIFIED CHARACTER ACCURACY FOR CARE UNDER DIFFERENT ℓ_2 PERTURBATION RADII ON WORD50.

σ	Method	ACR	Certified Accuracy under Radius r												
			0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.10	1.20
0.12	CARE (Gaussian)	0.306	86.6	80.6	71.8	60.4	50.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE (SmoothAdv)	**0.341**	**90.2**	**85.2**	**78.0**	**70.8**	**60.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE (Consistency)	0.318	86.6	80.8	72.8	65.6	53.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.25	CARE (Gaussian)	0.467	84.0	77.8	70.8	63.4	56.0	49.0	41.4	30.2	24.6	12.8	0.0	0.0	0.0
	CARE (SmoothAdv)	**0.539**	86.4	82.8	77.2	70.6	**63.2**	**55.6**	**50.2**	**41.8**	**32.4**	**21.2**	0.0	0.0	0.0
	CARE (Consistency)	0.522	**87.6**	**83.2**	**77.4**	**73.0**	62.8	54.4	46.2	37.0	29.0	17.8	0.0	0.0	0.0
0.50	CARE (Gaussian)	0.536	80.2	**76.8**	**70.4**	64.6	**59.2**	51.6	**44.8**	**37.0**	29.4	**21.6**	15.6	10.4	6.0
	CARE (SmoothAdv)	0.501	76.8	71.6	65.6	60.6	53.6	45.8	41.0	35.6	27.6	20.4	15.4	10.8	**6.6**
	CARE (Consistency)	**0.539**	**80.6**	75.6	70.2	**65.4**	**59.2**	**53.0**	44.6	36.2	**29.6**	**21.6**	**17.2**	**11.6**	6.4

$2,000$ different word images, respectively. We randomly select 10 images for each word from the test dataset for certification, and the total number of certified images is 500 following the standard evaluation setting. All the results are certified with $N = 100,000$ samples of smoothing noise, and the confidence of the certification is set to 99.9%. We test our method on three levels of smoothing noise $\sigma = 0.12, 0.25, 0.50$, and the η is set to $0.6, 0.9, 1.0$, respectively.

GTSRB certification details: The whole dataset contains 14880 training samples, 972 validation samples, and 3888 testing samples. We randomly pick one out of every eight from the test dataset for certification, and following the standard setting [13], we certify these 486 images with confidence 99.9%, and all the results are certified with $N = 100,000$ samples of smoothing noise. We test our method on three levels of smoothing noise with $\sigma = 0.12, 0.25, 0.50$, and the η is set to $0.10, 0.15, 0.25$, respectively.

PDF Malware certification details: We split the whole Contagio dataset into 70% train set and 30% test set following [48]. In specific, the number of malicious PDFs for training and testing is $6,896$ and $3,448$, respectively, while the number of benign PDFs for training and testing is $6,296$

TABLE XI
CERTIFIED ACCURACY UNDER DIFFERENT ℓ_2 PERTURBATION RADII AND SIGMA ON GTSRB.

σ	Method	ACR	Certified Accuracy under Radius r												
			0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.10	1.20
0.12	Gaussian	0.410	97.9	96.5	92.6	86.8	79.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SWEEN	0.417	99.2	97.1	94.7	87.9	82.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SmoothAdv	0.410	97.1	95.9	91.8	86.6	81.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Consistency	0.422	99.4	98.8	95.3	90.7	83.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	MultiTask	0.402	96.5	94.4	90.3	85.8	78.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE (Gaussian)	0.414	99.0	98.1	92.8	87.2	82.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE (SmoothAdv)	0.421	99.0	98.6	94.4	90.5	81.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE (Consistency)	**0.425**	**99.6**	**99.2**	**96.7**	**91.2**	**84.4**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.25	Gaussian	0.742	96.5	94.7	90.3	85.6	82.5	78.6	74.3	68.7	63.2	55.3	0.0	0.0	0.0
	SWEEN	0.750	97.7	94.2	90.7	86.6	82.9	78.8	74.1	69.1	65.8	58.4	0.0	0.0	0.0
	SmoothAdv	0.754	93.8	91.6	90.1	86.6	82.9	78.6	**75.3**	71.0	66.9	**63.0**	0.0	0.0	0.0
	Consistency	0.755	95.5	93.8	91.8	87.7	83.3	78.0	74.3	71.0	65.6	59.9	0.0	0.0	0.0
	MultiTask	0.732	95.7	93.4	90.3	85.8	81.3	76.3	71.8	66.9	62.8	54.3	0.0	0.0	0.0
	CARE (Gaussian)	0.754	**97.9**	**95.1**	91.2	86.2	82.9	79.0	75.1	70.4	65.2	58.6	0.0	0.0	0.0
	CARE (SmoothAdv)	**0.762**	95.5	93.8	91.4	87.0	**83.7**	**79.4**	75.1	**72.0**	**67.9**	**63.0**	0.0	0.0	0.0
	CARE (Consistency)	0.761	96.9	94.7	**92.2**	**87.7**	82.9	78.4	**75.3**	71.8	66.9	60.5	0.0	0.0	0.0
0.50	Gaussian	1.058	88.1	86.2	82.9	78.6	75.3	71.6	70.0	64.6	60.5	57.0	53.3	50.4	47.7
	SWEEN	1.092	87.9	86.0	**83.3**	79.6	75.5	72.8	69.5	66.3	63.2	58.6	55.8	52.5	49.6
	SmoothAdv	1.079	79.8	78.8	76.1	73.7	71.2	68.7	66.3	63.2	61.1	59.3	56.8	53.9	51.6
	Consistency	1.098	82.7	81.1	79.0	76.5	74.9	73.3	**70.8**	67.5	**64.6**	60.5	57.4	54.5	51.2
	MultiTask	1.082	89.5	87.2	82.5	78.4	75.5	72.8	69.8	67.3	62.8	57.8	55.3	51.4	48.6
	CARE (Gaussian)	1.092	**89.5**	**86.8**	82.9	**80.2**	76.3	73.3	69.3	66.3	62.6	58.6	55.3	52.7	49.4
	CARE (SmoothAdv)	1.111	87.9	84.6	81.7	78.2	74.7	72.0	70.0	65.8	63.0	**60.9**	57.2	54.3	51.6
	CARE (Consistency)	**1.117**	88.5	86.0	81.9	78.6	**76.5**	**73.5**	70.6	**67.7**	64.0	60.7	**57.8**	**55.1**	**51.9**

and $2,698$, respectively. We select 10% images, namely, 615 PDFs, from the test dataset for certification. Since the input extracted features from the PDF are binary, the certification is conducted based on ℓ_0-norm. All the results are certified with $N = 100,000$ samples of smoothing noise, and the confidence of the certification is set to 99.9%. We test our method on three levels of retaining probability $\alpha = 0.80, 0.85, 0.90$, and the η is set to $0.09, 0.10, 0.05$, respectively. All the sensors for both our method and the baselines are trained with Bernoulli augmentation following [38], where each feature value will be replaced with a random value {0,1} with probability $1 - \alpha$.

4) Detailed knowledge used in GTSRB: We demonstrate the 20 manually constructed attributes here. Some of them are adapted from [21]. The 12 permissive attributes are as follows: *"Octagon", "Square", "Blank Triangle", "Inverse Triangle", "Red Circle", "Gray Circle", "Blank Circle", "Digit 20", "Digit 50", "Digit 120", "Left Arrow", "Right Arrow"*. The 8 preventative attributes are as follows: *"Red Hollow Circle", "Blue Filled Circle", "Circle", "Blank Content", "Digit Content", "Filled Content", "Symmetric", "Polygon"*.

The inclusion relations are shown as follows: 1. each of the attributes *"Octagon", "Square", "Blank Triangle", "Inverse Triangle"* would imply *"Polygon"*; 2. each of the attributes *"Red Circle", "Gray Circle", "Blank Circle", "Red Hollow Circle", "'Blue Filled Circle"* would imply *"Circle"*; 3. each of the attributes *"Blank Triangle", "Blank Circle"* would imply *"Blank Content"*; 4. each of the attributes *"Digit 20", "Digit 50", "Digit 120"* would imply *"Digit Content"*. For better message passing, the edge for the inclusion relation on the graph is directed from the attribute to its corresponding implied attribute.

5) The knowledge and the reasoning details in PDF malware: The chosen 6 malicious traces are *"/Root/OpenAction", "/Root/OpenAction/S", "/Root/OpenAction/JS", "/Root/OpenAction/JS/Filter", "/Root/OpenAction/JS/Length", "/Root/OpenAction/Type"*. The chosen 8 benign traces are *"/Root/Metadata", "/Root/Metadata/Length", "/Root/Metadata/Subtype", "/Root/Metadata/Type", "/Root/Pages/Contents", "/Root/Pages/Contents/Filter", "/Root/Pages/Contents/Length", "/Root/Pages/CropBox"*.

For the reasoning part, notice we construct the formula like $t_i \implies \neg t_j$. Then, instead of directly connecting the edge between the node representing t_i and the node representing t_j, we construct an auxiliary node representing the predicate $\neg t_j$ and choose to connect it with the node representing t_i for better message passing. And the exclusion formula is naturally built for t_j and $\neg t_j$.

C. Detailed Experiment Results for CARE

We demonstrate the detailed experiment results for our method CARE with different main sensors. For simplicity, we use CARE (Gaussian) to indicate the main sensor is

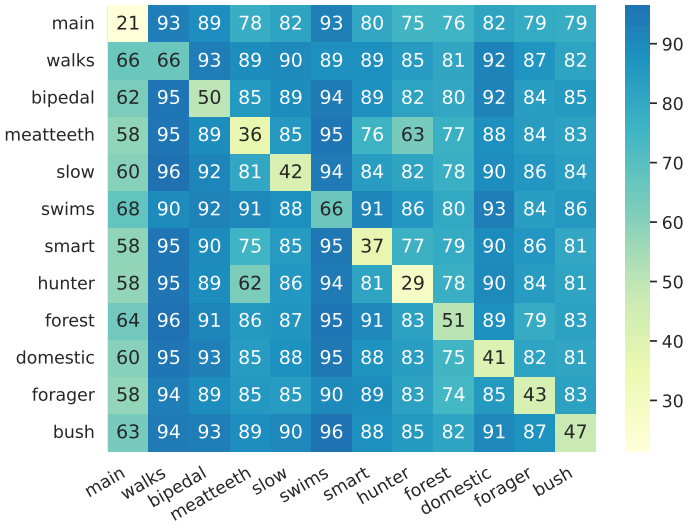


Fig. 6. The heatmap for the attack transferability between 13 sensors. The number in the cell (i, j) represents the empirical robust accuracy (%) of jth sensor when tested with the adversarial attacks against the ith sensor.

trained with vanilla Gaussian augmentation [13]; use CARE (SmoothAdv) to indicate the main sensor is trained with adversarial training [35]; and use CARE (Consistency) to indicate the main sensor is trained with consistency regularization [36]. Notice that all the knowledge sensors are trained with vanilla Gaussian augmentation [13]. The detailed results on AwA are shown in Table VIII; the results for word classification and character classification are shown in Table IX and Table X, respectively.

The full results for GTSRB are shown in Table XI; while for the PDF malware dataset, we also report the Median Certified Radius (MCR) as a reference, and the full results are shown in Table XII.

D. Exploration for the importance of the reasoning module

In this section, we explore the importance of the reasoning module by simply replacing the GCN in CARE with a linear classifier, which learns to discriminate along both the main sensor and knowledge sensors directly without a reasoning component. For simplicity, we denote this method as *SensingLinear* and similarly, we also train it with different main sensors, including Gaussian, SmoothAdv, and Consistency; the final certified results are shown in Table XIII. As we can see, although both main and knowledge sensors are aggregated in this linear classifier, the performance will still drop significantly without the reasoning part, which demonstrates the necessity of the construction of logical reasoning based on the output of both the main sensor and the knowledge sensors.

E. Increasing the number of the base models for SWEEN

We provide the experiment results for increasing the number of base models for SWEEN on AwA2. The magnitude of the smoothing noise is set to 0.50 here, and we certify the ensemble method SWEEN with the number of base model $m \in \{3, 6, 10, 15, 20, 30\}$, the corresponding result is shown in Table XIV. As we can see, the performance improvement from the increase of the base model is marginal, and the certified accuracy only improves a bit ($1 \sim 2\%$) when we increase the base models from 3 to 30. On the contrary, with the domain knowledge and logic, such a phenomenon is alleviated as shown in Table VI with a $10 \sim 30\%$ improvement.

F. Additional Experiment Results and Details for Empirical Attack

1) Experiment details: We implement the empirical untargeted attack as follows: (1.) Take the mean of the output confidence from the soft base main sensor for the corrupted input image with 100 Gaussian noise; (2.) Use projected gradient descent (PGD) [79] to minimize the mean confidence for the truth label and get the corresponding adversarial image; (3.) Next, this adversarial image will be sent to all the knowledge sensors to get the new adversarial sensing vector $\boldsymbol{z}'$; (4.) Do the same hypothesis-test-based prediction procedure in [13] with our method CARE to check if the attack is successful with $\boldsymbol{z}'$. The test images here are the same as those in the certification part. For ℓ_∞ attack, the number of update steps is also fixed to 40, the attack step size is set to $1/255$, and the full results are shown in Table XV.

2) Attack transferability: Based on these test images, we also explore the attack transferability between 12 sensors (one main sensor and eleven random picked attribute sensors, all are trained under $\sigma = 0.50$) on AwA2 under ℓ_2 perturbation size $\epsilon = 3.0$. Besides, the attack step size is set to 0.2, the number of update steps is set to 100, and the final results are shown in Figure 6.

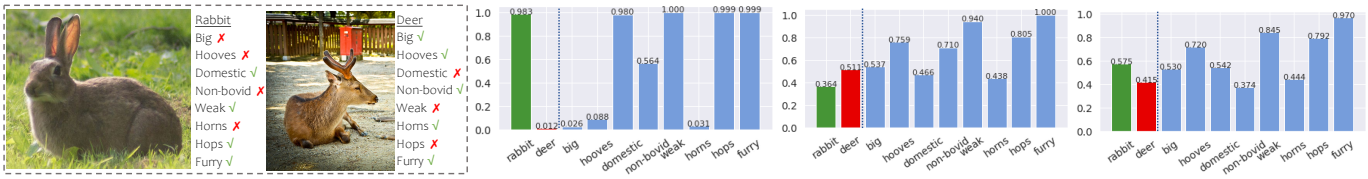
3) Attacking with the attributes: We also conduct the experiments which construct the adversarial image by attacking the main sensor and all the attribute sensors at the same time on Word50. In other words, the PGD attack is implemented to increase the mean of the loss from both the word and letter classifications. In this case, we also report the empirical robust accuracy of the main sensor (Gaussian/SmoothAdv/Consistency) used in CARE on these new adversarial images, and the results are shown in Table XVI. As we can see, even though the attack seems stronger here, with the incorporation of the domain knowledge, our method is still much more robust than the baselines.

G. Case Study on AwA2

In this section, we provide more case studies on AwA2, which are shown in Figures 7 to 11. In specific, we adopt untargeted attacks here; the ℓ_2 perturbation size is set to 3.0, the attack step size is set to 0.2, the number of update steps is set to 100, and all the sensors used here are trained under $\sigma = 0.50$.

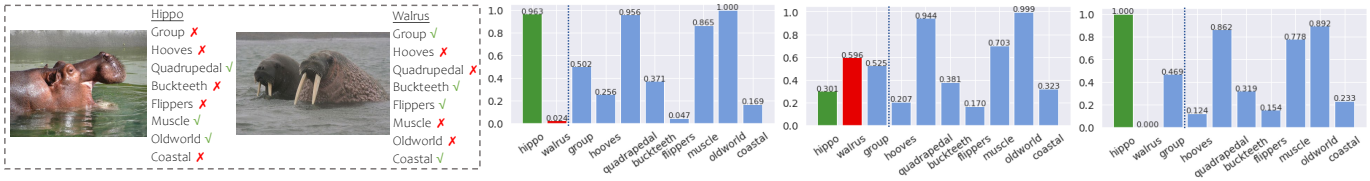
TABLE XII
CERTIFIED ACCURACY UNDER DIFFERENT RETAINING PROBABILITY α AND ℓ_0 PERTURBATION RADII ON THE PDF MALWARE DATASET.

α	Method	ACR/MCR	Certified Accuracy under Radius r									
			0	1	2	3	4	5	6	7	8	9
0.90	Lee et al. [38]	3.008/3	**99.8**	99.0	96.1	77.9	27.8	0.0	0.0	0.0	0.0	0.0
	SWEEN	3.159/3	**99.8**	99.0	**97.7**	81.1	38.0	0.0	0.0	0.0	0.0	0.0
	MultiTask	3.341/4	99.7	99.0	97.2	81.5	56.4	0.0	0.0	0.0	0.0	0.0
	CARE	**3.506/4**	99.5	**99.3**	96.9	**85.5**	**68.8**	0.0	0.0	0.0	0.0	0.0
0.85	Lee et al. [38]	3.842/4	**99.7**	98.5	96.1	80.0	53.5	43.7	12.4	0.0	0.0	0.0
	SWEEN	4.367/5	**99.7**	**98.9**	**96.4**	82.2	68.6	65.4	22.3	0.0	0.0	0.0
	MultiTask	4.626/6	99.7	99.0	96.7	82.8	66.3	65.0	52.7	0.0	0.0	0.0
	CARE	**4.954/6**	99.5	**98.9**	91.1	**85.4**	**79.3**	**77.4**	**63.4**	0.0	0.0	0.0
0.80	Lee et al. [38]	4.945/5	**99.5**	98.7	94.8	80.0	80.0	68.0	46.5	15.1	5.7	5.7
	SWEEN	5.259/6	**99.5**	**98.9**	95.8	80.7	80.3	72.5	57.2	22.6	8.9	8.9
	MultiTask	5.624/7	99.3	98.7	96.1	81.8	80.5	72.7	59.0	53.8	9.9	9.9
	CARE	**5.789/7**	99.2	98.4	**96.6**	**84.2**	**84.2**	**74.5**	**59.5**	**54.5**	**13.5**	**13.5**



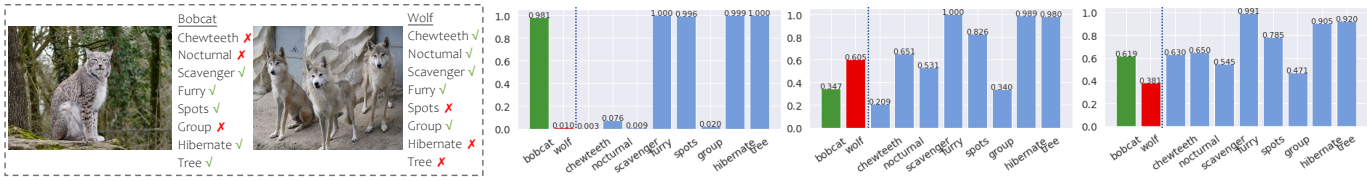
(a) (b) (c) (d)

Fig. 7. The illustration of the change of confidence. (a) the attributes for the rabbit and mole; (b) the original confidence before the attack; (c) the confidence for the adversarial image, which is obtained by attacking the main sensor; (d) the recovered confidence from our method CARE for the adversarial image. The ground truth is *"rabbit"*.



(a) (b) (c) (d)

Fig. 8. The illustration of the change of confidence. (a) the attributes for the hippopotamus and walrus; (b) the original confidence before the attack; (c) the confidence for the adversarial image, which is obtained by attacking the main sensor; (d) the recovered confidence from our method CARE for the adversarial image. The ground truth is *"hippopotamus"*.



(a) (b) (c) (d)

Fig. 9. The illustration of the change of confidence. (a) the attributes for the bobcat and wolf; (b) the original confidence before the attack; (c) the confidence for the adversarial image which is obtained by attacking the main sensor; (d) the recovered confidence from our method CARE for the adversarial image. The ground truth is *"bobcat"*.

TABLE XIII

CERTIFIED ACCURACY FOR THE METHOD *SensingLinear* ON THE AWA2 DATASET UNDER DIFFERENT ℓ_2 PERTURBATION RADII, AND THE USED MAIN SENSOR IS INDICATED IN THE BRACKET.

σ	Method	ACR	Certified Accuracy under Radius r												
			0.00	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00	2.20	2.40
0.25	SensingLinear(Gaussian)	0.593	79.8	78.2	76.2	71.0	58.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SensingLinear(SmoothAdv)	0.596	80.2	78.4	76.6	71.0	60.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SensingLinear(Consistency)	0.596	79.4	78.2	76.4	72.2	60.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CARE	**0.709**	**96.6**	**94.2**	**91.4**	**85.4**	**75.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.50	SensingLinear(Gaussian)	0.842	69.6	67.6	63.2	58.2	53.4	49.4	42.4	36.8	27.2	0.0	0.0	0.0	0.0
	SensingLinear(SmoothAdv)	0.865	70.0	67.6	64.2	59.6	55.2	50.4	45.2	40.0	31.6	0.0	0.0	0.0	0.0
	SensingLinear(Consistency)	0.877	69.8	68.0	65.4	60.0	56.0	51.0	45.6	40.4	31.6	0.0	0.0	0.0	0.0
	CARE	**1.141**	**91.2**	**88.2**	**84.2**	**78.8**	**73.4**	**68.4**	**63.2**	**56.2**	**44.0**	0.0	0.0	0.0	0.0
1.00	SensingLinear(Gaussian)	1.192	51.6	49.8	48.4	46.8	46.0	45.0	42.0	40.0	38.2	36.0	34.0	31.2	29.2
	SensingLinear(SmoothAdv)	1.265	52.8	51.6	50.2	48.4	47.4	46.4	45.0	42.2	40.0	37.8	36.0	34.2	32.0
	SensingLinear(Consistency)	1.270	52.2	51.4	50.2	48.8	47.4	46.4	44.0	43.2	40.6	38.6	36.4	34.2	32.4
	CARE	**2.127**	**87.0**	**85.2**	**84.0**	**82.0**	**80.4**	**78.2**	**75.6**	**71.4**	**68.6**	**65.8**	**61.8**	**59.4**	**56.0**

TABLE XIV

CERTIFIED ACCURACY FOR SWEEN WITH DIFFERENT NUMBER OF BASE MODELS UNDER SMOOTHING NOISE LEVEL $\sigma = 0.50$.

# base models	ACR	Certified Accuracy under ℓ_2 Radius r								
		0.00	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60
$m = 3$	0.846	**76.8**	72.8	66.6	60.2	53.8	46.4	39.0	34.4	22.4
$m = 6$	0.854	76.4	**73.8**	67.8	60.4	53.6	47.4	39.6	34.6	22.4
$m = 10$	0.856	76.4	73.4	**68.2**	59.8	53.6	47.8	40.6	34.6	**23.6**
$m = 15$	0.859	76.4	73.0	**68.2**	59.4	53.4	47.8	40.4	35.0	**23.6**
$m = 20$	0.860	76.6	72.8	**68.2**	59.4	53.6	48.0	40.6	35.0	23.4
$m = 30$	**0.863**	**76.8**	73.2	67.4	**60.8**	**54.4**	**48.2**	**41.2**	**35.2**	23.4

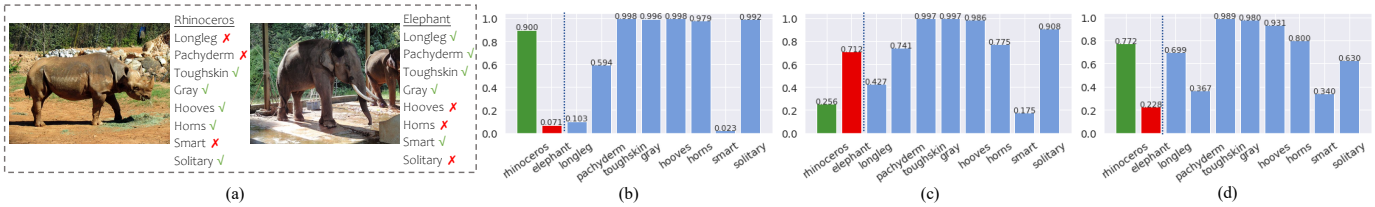


Fig. 10. The illustration of the change of confidence. (a) the attributes for the rhinoceros and elephant; (b) the original confidence before the attack; (c) the confidence for the adversarial image, which is obtained by attacking the main sensor; (d) the recovered confidence from our method CARE for the adversarial image. The ground truth is *"rhinoceros"*.

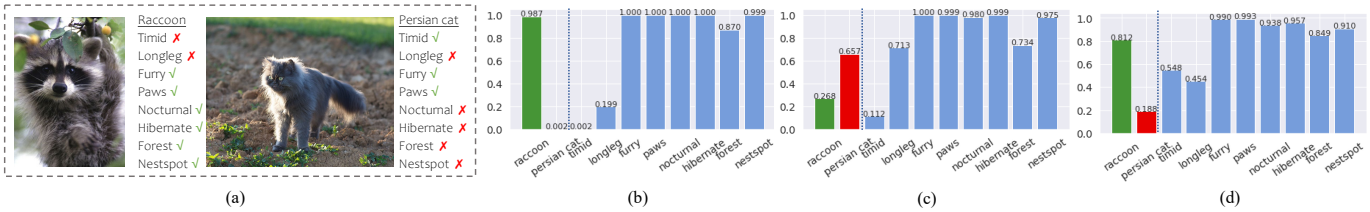


Fig. 11. The illustration of the change of confidence. (a) the attributes for the raccoon and Persian cat; (b) the original confidence before the attack; (c) the confidence for the adversarial image, which is obtained by attacking the main sensor; (d) the recovered confidence from our method CARE for the adversarial image. The ground truth is *"raccoon"*.

TABLE XV
THE EMPIRICAL ROBUST ACCURACY OF DIFFERENT METHODS FOR AWA2, WORD50 AND GTSRB UNDER ℓ_∞ METRIC.

Method	AwA2				Word50				GTSRB			
	σ	ϵ			σ	ϵ			σ	ϵ		
		2/255	4/255	8/255		2/255	4/255	8/255		2/255	4/255	8/255
Gaussian		42.4	8.6	0.0		11.4	1.4	0.0		89.9	73.7	47.3
SWEEN		45.8	10.6	0.4		30.4	13.4	2.6		94.2	**84.2**	65.6
SmoothAdv	0.25	53.6	23.0	1.6	0.12	32.4	10.4	1.2	0.12	87.2	74.7	56.6
Consistency		45.6	13.6	0.4		22.6	5.6	0.0		92.4	79.0	57.2
CARE		**80.6**	**32.6**	**3.4**		**87.0**	**83.0**	**81.2**		**94.7**	84.0	**68.7**
Gaussian		47.0	18.2	1.6		14.4	2.6	0.0		86.2	75.1	51.9
SWEEN		47.6	21.6	2.4		30.0	18.2	2.8		87.4	76.7	58.4
SmoothAdv	0.50	48.6	25.8	3.4	0.25	30.8	13.6	1.0	0.25	85.6	74.5	57.4
Consistency		48.4	25.0	3.0		25.6	10.8	0.4		86.4	75.7	56.6
CARE		**72.6**	**44.8**	**8.6**		**90.0**	**85.4**	**79.4**		**88.7**	**78.2**	**61.3**
Gaussian		40.4	22.0	4.4		12.8	4.2	0.2		78.0	67.9	48.1
SWEEN		41.6	25.0	6.8		21.4	12.8	3.6		**78.4**	69.1	51.4
SmoothAdv	1.00	39.8	25.6	6.6	0.50	22.6	10.8	2.2	0.50	72.6	65.2	49.8
Consistency		40.0	27.2	9.2		18.0	9.4	1.4		75.5	70.8	52.1
CARE		**78.8**	**66.8**	**37.2**		**81.0**	**74.2**	**63.0**		**78.4**	**72.4**	**57.4**

TABLE XVI
THE EMPIRICAL ROBUST ACCURACY OF DIFFERENT METHODS FOR WORD50 WHEN ATTACKING ON BOTH WORD CLASSIFICATION AND LETTER CLASSIFICATION UNDER DIFFERENT METRICS INCLUDING ℓ_2 AND ℓ_∞.

Method	σ	$\epsilon(\ell_2)$			$\epsilon(\ell_\infty)$		
		0.4	0.8	1.2	2/255	4/255	8/255
Gaussian		22.8	16.0	12.0	18.0	7.0	3.2
SmoothAdv		46.8	40.2	34.0	42.2	29.8	15.8
Consistency	0.12	34.8	26.6	22.6	29.2	17.8	9.2
CARE (Gaussian)		81.0	65.6	47.4	83.4	69.8	40.6
CARE (SmoothAdv)		81.8	65.2	49.6	83.0	65.8	30.6
CARE (Consistency)		**85.4**	**69.2**	**52.6**	**90.2**	**78.6**	**44.2**
Gaussian		21.4	11.4	7.0	18.0	7.2	2.0
SmoothAdv		39.0	31.2	25.0	37.6	27.2	12.8
Consistency	0.25	33.2	26.0	19.8	31.6	20.8	8.6
CARE (Gaussian)		84.0	69.6	54.4	86.2	76.2	**55.4**
CARE (SmoothAdv)		77.0	62.6	48.2	79.2	66.8	34.8
CARE (Consistency)		**88.6**	**74.2**	**58.6**	**92.0**	**81.4**	53.8
Gaussian		17.0	8.2	4.2	16.2	7.2	1.8
SmoothAdv		25.6	16.6	11.4	27.4	18.4	8.0
Consistency	0.50	14.0	7.8	4.8	23.4	14.0	6.8
CARE (Gaussian)		73.6	**61.4**	45.2	76.8	**66.6**	42.4
CARE (SmoothAdv)		69.6	56.8	43.0	73.6	62.4	39.8
CARE (Consistency)		**74.0**	60.4	**46.2**	**77.6**	65.8	**45.0**