CARD: CERTIFIABLY ROBUST MACHINE LEARNING PIPELINE VIA DOMAIN KNOWLEDGE INTEGRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The advent of ubiquitous machine learning (ML) has led to an exciting revolution in computing today. However, recent studies have shown that ML, especially deep neural networks (DNNs), are vulnerable to adversarial examples, which are able to mislead DNNs with carefully crafted stealthy perturbations. So far, many defense approaches have been proposed against such adversarial attacks, both empirically and theoretically. Though effective under certain conditions, existing empirical defenses are usually found vulnerable against new attacks; existing certified defenses are only able to certify robustness against limited perturbation radius. As current *pure data-driven* defenses have reached a bottleneck towards certifiably robust ML, in this paper we propose a certifiably robust ML pipeline CARD, aiming to integrate exogenous information, such as domain knowledge, as logical rules with ML models to improve the certified robustness. Intuitively, domain knowledge (e.g., the cat belongs to the mammal category) will prevent attacks that violate these knowledge rules, and it is also challenging to construct adaptive attacks satisfying such pre-defined logic relationships. In particular, we express the domain knowledge as the first-order logic rules and embed these logic rules in a probabilistic graphical model. We then prove that such a probabilistic graphical model can be mapped to a 1-layer NN for efficient training. We conduct extensive experiments on several high-dimensional datasets and show that our proposed CARD achieves state-of-the-art certified robustness.

1 INTRODUCTION

Despite the great advances achieved by deep neural networks (DNNs), recent studies show that DNNs are vulnerable to carefully crafted adversarial perturbations, which are usually of small magnitude while being able to mislead the predictions arbitrarily (Biggio et al., 2013; Szegedy et al., 2014; Xiao et al., 2018a;b). As machine learning techniques are incorporated into safety-critical systems—from financial systems to self-driving cars to medical diagnosis—it is vitally important to develop robust learning approaches before massive production and deployment of safety-critical ML applications such as autonomous driving. To improve the robustness of machine learning models, several empirical and certified defenses have been proposed against such attacks. However, most existing *empirical defenses* have been attacked again by strong adaptive attackers (Carlini and Wagner, 2017; Athalye et al., 2018); most *certified defenses* are restricted to certifying the model robustness within a small ℓ_p norm bounded perturbation radius (Salman et al., 2019b; Yang et al., 2020). One potential limitation for existing robust learning approaches is inherent in the fact that most of them have been treating machine learning as a "pure data-driven" technique that solely depends on a given training set, without interacting with the rich exogenous information such as domain knowledge; while we know human, who has knowledge and inference abilities, is resilient to such attacks. Thus, in this paper we aim to explore the questions: *Can we incorporate human knowledge to DNNs to improve their robustness? Can we provide certified robustness for such knowledge enabled machine learning pipelines?*

Intuitively, there is information from the open world that could help with ML robustness: E.g., "*persian cat is usually long-haired, furry, with whisker and tail*". Integrating such extrinsic information enables the system to have the capacity of *common sense reasoning*, and therefore correct mispredictions that do not satisfy such prior knowledge. In addition, such systems will also be hard for attackers to conduct adaptive attacks which need to satisfy these knowledge rules. To efficiently integrate knowledge into DNNs and be able to certify the learning pipeline, we propose the first

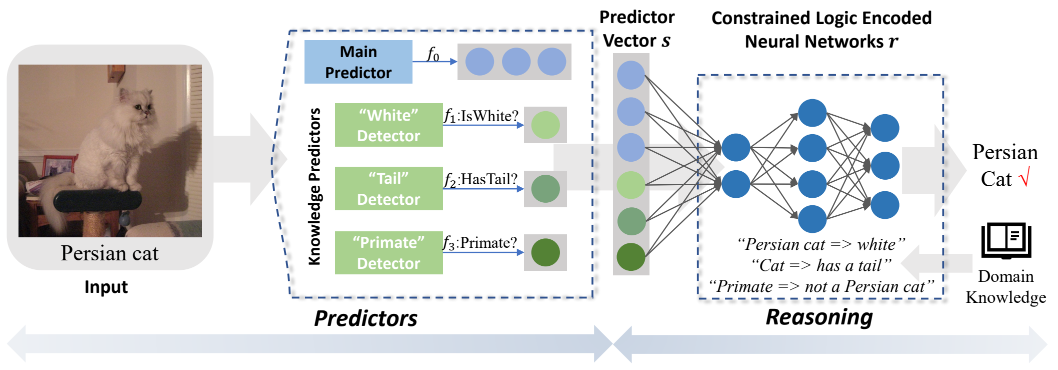

Figure 1: An overview of the CARD framework.

unified certifiably robust machine learning pipeline via domain knowledge integration (CARD). In particular, CARD contains a main predictor, several knowledge predictors, and a reasoning component as shown in Figure 1. The main predictor makes an overall prediction for a given input, each knowledge predictor makes binary "knowledge prediction" ($e.g.$, whether the input is a cat), and the reasoning component aims to build logical relationships among the outputs of predictors.

Concretely, we express domain knowledge as the first-order logical rules ($e.g.$, persian cat $\implies$ cat), which will build connections between different predictors; we then leverage the soft logic (Bach et al., 2017) as relaxation to encode the binary outputs of knowledge predictors as $[0, 1]$ ($e.g.$, prediction confidence) and encode these logical rules as matrices. Finally, we transform the matrices into a one-layer neural network as the reasoning component. Note that we can also build different probabilistic graphical models such as Markov logic networks as the reasoning component, but the inference for such models is #P-Complete. We prove that such a reasoning process can be mapped as a one-layer neural network, which is computationally efficient. We also prove that the certified robustness of CARD is higher than that of a standard weighted ensemble or a single model.

To evaluate CARD, we conduct thorough experiments on large-scale datasets: Animals with Attributes (AwA2) (Xian et al., 2018), and Word50 (Chen et al., 2015). For AwA2, we leverage the annotated attributes of each class and the hierarchy relationship between classes as our knowledge. For Word50, we leverage the positions of characters in the known word set as knowledge. We show that CARD significantly outperforms the STOA certified defenses (Salman et al., 2019a; Jeong and Shin, 2020; Liu et al., 2020) under different perturbation radii.

Technical Contributions. In this paper, we provide the *first* machine learning framework CARD on improving the certified robustness via knowledge integration. We make contributions on both theoretical and empirical fronts.

- We propose the first certifiably robust learning framework CARD with knowledge integration.
- We prove that the reasoning process based on logic rules can be mapped as a one-layer neural network. We also prove that the certified robustness of CARD is theoretically higher than that of a standard weighted ensemble or a single model.
- We conduct extensive experiments on different datasets to show that CARD achieves *significantly* higher certified robustness than SOTA baselines. For instance, on the word-level classification on Word50, under ℓ_2 radius 0.5, CARD improves the certified accuracy from 15.4% (SOTA) to 53.6%; and on AwA2 under ℓ_2 radius 1.5, CARD improves the certified accuracy from 43.8% (SOTA) to 65.4%. We also provide several ablation studies to explore the utility of different knowledge, the impact of the number of predictors, and relaxation strategies for the reasoning component.

2 Related Work

Knowledge integration and logical reasoning. Deng et al. (2014) proposed the hierarchy and exclusion graphs to utilize the hierarchy relations between different classes in ImageNet for classification tasks. Yan et al. (2015) tried to embed deep convolutional neural networks into a two-level category hierarchy to improve the prediction accuracy on ImageNet. The hierarchical knowledge among classes has been incorporated into the training to avoid worst-case failures (Bertinetto et al., 2020). On the other hand, some work has applied the declarative logic to represent domain knowledge for training different models. The expressive power of logic programming and the generalizations of the first-logic rules including Horn clauses have been studied extensively by Chandra and Harel (1985); Dantsin et al. (2001). With the combination of neural networks and logic reasoning, Hu et al. (2016) transfers the first-logic rules into weights of neural networks by distillation. However, existing work mainly relies on using different knowledge to improve the benign accuracy of a machine learning model, while our work focuses on improving its certified robustness via knowledge integration.

Certified robustness There are two sub-branches for the certified robustness approaches. The first is the *complete certification*, which aims to provide complete certification for each instance on whether it can be certified within a certain l_p norm radius. However, the complete certification for the commonly used feed-forward ReLU networks is NP-complete, and it is usually heuristically optimized based on either satisfiability modulo theories (Katz et al., 2017; Ehlers, 2017), or mixed integer-linear programming (Lomuscio and Maganti, 2017; Fischetti and Jo, 2017), which are computationally expensive. The other branch is *incomplete certification*, which aims to provide certification with certain relaxation. *Randomized smoothing* is proposed to certify large-scale datasets such as ImageNet (Cohen et al., 2019). Several following works are proposed to further improve the certification bounds: Salman et al. (2019a) proposed to integrate adversarial training to further improve the robustness certification; a consistency regularization is applied during training to improve the certification (Jeong and Shin, 2020). Based on existing certified defenses, we design the knowledge-enhanced certifiably robust machine learning pipeline and certify it with standard certification.

3 CARD PIPELINE

In this section, we will first describe the building blocks of CARD, and then discuss the details of the training and certification process of CARD, followed by our theoretical analysis for CARD.

Overview of CARD. To effectively integrate domain knowledge into machine learning pipelines, we propose CARD, which consists of three parts: *main predictor, knowledge predictor, and reasoning component*. In particular, the main predictor servers for the main classification task and makes the final multi-class prediction given an input. The knowledge predictors make predictions for the individual objects within a knowledge rule based on the same input. For instance, if we want to integrate the knowledge "persian cat belongs to cat", we will train two knowledge predictors to predict whether the input is a persian cat and whether it is a cat, respectively. We then represent the knowledge as first-order logic rule "persian cat $\implies$ cat" via a probabilistic graphical model and obtain the weight matrix of the model. Finally, we map the weight matrix as a one-layer neural network, which takes the output of all main and knowledge predictors as inputs and makes the final classification prediction based on the calculated weighted penalty scores of each knowledge rule. Since this pipeline can be viewed as a general machine learning classifier composing several neural networks, we are able to certify its robustness using standard certification approaches (Cohen et al., 2019). In this paper, we consider different types of domain knowledge such as attributes-based knowledge and category hierarchy knowledge to improve the certified robustness (details in Section 4).

Note that the CARD pipeline is quite flexible, and its robustness can be certified and compared fairly with the state-of-the-art robust training algorithms (Salman et al., 2019a; Jeong and Shin, 2020). 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, we show that the proposed CARD is able to *significantly* improve the certified robustness on datasets including the high-resolution data AwA2 (Xian et al., 2018) and standard Word50 (Chen et al., 2015). In addition, we will also show both theoretically and empirically that the robustness improvement indeed comes from the knowledge integration rather than simply adding more prediction models as an ensemble, which is shown to obtain marginal robustness improvement (Liu et al., 2020; Yang et al., 2021). We believe such knowledge integration is a promising way to break the current robustness barriers.

3.1 BUILDING THE CARD PIPELINE

We will next introduce each component of CARD, and the intuition for its robustness improvement.

Main and Knowledge Predictors. We first train a main predictor: $f_0 : \mathbb{R}^d \to \mathcal{Y}_0$, together with several knowledge predictors: $f_i : \mathbb{R}^d \to \mathcal{Y}_i$ where $i \in \{1, 2, \ldots, n\}$. Here $\mathcal{Y}_0$ denotes the main classification prediction and each $\mathcal{Y}_i$ corresponds to a subtask related to the knowledge. The output of all predictors would be concatenated into one *predictor vector* $\boldsymbol{s} \in [0, 1]^m$ as the input to the reasoning component $r : \mathbb{R}^m \to \mathcal{Y}_0$ ($m = \sum_{j=0}^{n} |\mathcal{Y}_j|$). Here we consider two types of knowledge.

• *Attribute-based Knowledge:* Every class would infer certain attributes, which can be represented as the knowledge rules. For instance, *cat* $\implies$ *has a tail* $\wedge$ *furry* $\wedge$ *quadrupedal*. To this end,

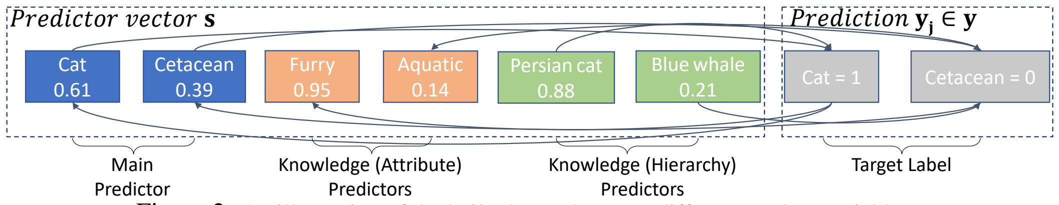

Figure 2: An illustration of the built clauses between different random variables.

the knowledge predictors would be trained to recognize each of such attributes, and the reasoning component would realize their relationships. Note that we can consider one or a subset of attributes.

• *Hierarchy Knowledge:* In general, different classes have hierarchical relationships with each other as specified in Deng et al. (2014). For instance, *persian cat* $\vee$ *bobcat* $\vee$ *siamese cat* $\implies$ *cat*. Thus, a knowledge predictor would be trained to classify each object, and the knowledge relationship between them will be controlled by the reasoning component.

Reasoning Component. Given the predictor vector $\boldsymbol{s}$ from all the predictors, to integrate domain knowledge, we will use a set of first-order logic rules to define the knowledge clauses, representing the logic relationship between one or a subset of dimensions of $\boldsymbol{s}$ with *prediction set* $\boldsymbol{y}$ as shown in Figure 2. We use s_i to represent the ith dimension of $\boldsymbol{s}$, and $\boldsymbol{y}_j$ as the jth one hot vector of the prediction set $\boldsymbol{y} = \{\boldsymbol{y}_j\}_{j=1}^{|\mathcal{Y}_0|}$. To character each logic clause C_l, we compute a *penalty score* p_l, which measures its inconsistency with prior knowledge rules, and clause *importance score* w_l.

Definition 1. (Clause Score). Let $\boldsymbol{C} = (C_1, \dots, C_L)$ represents a set of logic clauses, each of which is defined by the relationship between a subset of dimensions of $\boldsymbol{s}$ and $\boldsymbol{y}_j \in \boldsymbol{y}$. We characterize all the clauses by the penalty score vector $\boldsymbol{p}$ and importance score vector $\boldsymbol{w}$. The final *clause score* is the element-wise product of the two vectors: $\boldsymbol{u} = \boldsymbol{w} \cdot \boldsymbol{p}$.

During inference, the prediction $\boldsymbol{y}_j$ with the lowest summed clause score (weighted penalty) over all clauses would be the final output. In other words, the prediction will maximally satisfy the defined knowledge rules. Concretely, each clause C_l takes the vector $\boldsymbol{z} := [\boldsymbol{s}; \boldsymbol{y}_j]$ as input and outputs the corresponding penalty score p_l. Here we use the Łukasiewicz soft logic (Bach et al., 2017) to encode the logic rules for better optimization. The range of the truth values of each variable s_l is thus extended from $\{0, 1\}$ to $[0, 1]$. Then the Boolean logic operators are reformulated as:

$$\begin{aligned} &\neg a = 1 - a, \\ &a \,\&\, b = \max\{a + b - 1, 0\}, \\ &a \vee b = \min\{a + b, 1\}, \\ &a_1 \wedge \cdots \wedge a_n = \frac{1}{n}\sum_i a_i, \end{aligned} \tag{1}$$

The "averaging" operator $\wedge$ is a linear approximation for the conjunction operator $\&$ here, and will be used to replace the operator $\&$ following Beltagy et al. (2014); Foulds et al. (2015).

Basically, the truth value of relation "$x \implies y$" could be modeled by logic "$\neg x \vee y$". When the clause "$x \implies y$" is more likely to be true, the value of "$\neg x \vee y$" would be larger. To efficiently encode logical clauses into neural networks for better optimization, we propose a new Boolean logic operator $\sqcup$ to measure the degree of inconsistency between the clause and prior knowledge:

$$a \sqcup b = \max\{1 - a - b, 0\}. \tag{2}$$

As we can see, the degree of the inconsistency of "$x \implies y$" can be modeled by logic "$\neg x \sqcup y = \max\{1 - (1 - x) - y, 0\} = \max\{x - y, 0\}$", which is actually the prototype of ReLU activation.

Given a random variable t representing the target class and the predictor vector $\boldsymbol{s}$, we are able to encode four types of clauses and their inconsistency penalty scores as below:

$$\begin{aligned} &\text{Type 1: } t \implies s_i \vee s_j \vee ... \vee s_k : \neg t \sqcup (s_i \vee s_j \vee ... \vee s_k), \\ &\text{Type 2: } t \implies s_i \wedge s_j \wedge ... \wedge s_k : \neg t \sqcup (s_i \wedge s_j \wedge ... \wedge s_k), \\ &\text{Type 3: } s_i \vee s_j \vee ... \vee s_k \implies t : \neg(s_i \vee s_j \vee ... \vee s_k) \sqcup t, \\ &\text{Type 4: } s_i \wedge s_j \wedge ... \wedge s_k \implies t : \neg(s_i \wedge s_j \wedge ... \wedge s_k) \sqcup t, \end{aligned} \tag{3}$$

where $i, j, ..., k \in \{1, 2, ..., m\}$, which means each time we would pick a subset of predictors' outputs to build the clause. All the random variables above could also take negation, and the clauses like

$t = s_i$ built for the main predictor could be decomposed to two clauses $t \implies s_i$ and $s_i \implies t$ with the same importance score, which would be further discussed in Appendix A.

Example. Take the binary classification in Figure 2 as an example. Assume the ground truth label is *cat*. For simplification, here we only build two clauses: *cat* $\implies$ *furry*, *cetacean* $\implies$ *aquatic*. Thus, the penalty score of the first clause is $\max\{1 - 0.95, 0\} = 0.05$, which is a small penalty, and that of the second clause is $\max\{0 - 0.14, 0\} = 0$. If we assume the importance score of all clauses is 1, the clause score is 0.05. On the other hand, if we assume the target label is *cetacean*, the corresponding penalty scores of these two clauses are 0 and 0.86 respectively, and the total clause score is 0.86. As a result, we would pick the class with the minimal clause score as the final prediction, namely, *cat*. It is easy to see that if the adversarial attack aims to manipulate the main predictor to misrecognize *cat* as *cetacean*, as long as some of the knowledge predictors are correct, say the "furry" knowledge predictor is correct, these logical clauses would help to correct the final prediction.

Theorem 1 (1-NN Reasoning Representation)**.** *Given the predictor vector* $\boldsymbol{s}$*, the reasoning process of optimizing the prediction confidence for each class in* $\mathcal{Y}_0$ *based on the clause inconsistency penalty scores can be formed as the one layer neural networks:*

$$r(\boldsymbol{s}) = \underset{c \in \mathcal{Y}_0}{\arg\min} \ \left(\boldsymbol{W} \mathrm{ReLU}(\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B})) \right), \tag{4}$$

where W *is the matrix related to the importance score* $\boldsymbol{w}$*;* A *and* B *are the matrices determined by the overall logic relationships and related to the clause penalty score* $\boldsymbol{p}$*.*

Proof sketch. We decompose the proof into two parts. First, we show that the calculation of the allowed types of clauses in Equation (3) could be transformed to some linear functions of $\boldsymbol{z} = [\boldsymbol{s}; \boldsymbol{y}_j]$ with a ReLU activation, based on which we could get a temporary matrix and one bias vector. Then, we remove the iterative assignment of $\boldsymbol{y}_j$ from $\boldsymbol{z}$ by dividing this temporary matrix into two blocks: one is related to the multiplication with $\boldsymbol{s}$, and one is related to the multiplication with $\boldsymbol{y}_j$. Since all the assignments of the one-hot vectors in $\boldsymbol{y}$ could be concatenated as an identity matrix, the inconsistency penalty score for each class could be parallelly calculated by matrix multiplication, resulting in the final $\boldsymbol{A}$ and $\boldsymbol{B}$ shown here. The detailed proof deferred to Appendix A.1. We also provide an example in Appendix A.2 to illustrate this transformation process.

3.2 Training the CARD Pipeline

Training predictors. To certify the robustness of CARD, each predictor is trained separately with Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ augmentation. To boost the robustness of the knowledge predictors, they are designed as binary classification, and we use the *Binary Cross Entropy(BCE)* for training. In particular, based on the logic relationships, we can see that for attribute-based knowledge, the true positive rate is more important to correct the final prediction of the main predictor. Thus, when the predictor is trained for the attribute-based tasks, we would increase the loss weight of *BCE*$(f_i(x + \epsilon), 1)$ for the positive input x. Similarly, for hierarchy knowledge rules, the loss weight of *BCE*$(f_i(x + \epsilon), 0)$ would be increased for negative input x since the true negative rate for this type of knowledge is critical. Note that actually there is no need for extensive hyperparameter tuning in CARD, and in our experiment setting, we just set the increased loss weight to 2 and the rest as 1.

Pseudo-training for Importance scores of logic clauses. After we can calculate the penalty score of each clause by comparing it with given knowledge rules, we next describe how we train the reasoning component to obtain the importance score for each clause. Given the predictor vector $\boldsymbol{s}$, and the ground truth label y_{gt}, the importance score $\boldsymbol{w}$, which is encoded into the matrix $\boldsymbol{W}$ in Equation (4), is learned through minimizing the negative log-likelihood $-\ln \mathbb{P}[r(\boldsymbol{s}) = y_{\text{gt}}]$. The importance scores $\boldsymbol{w}$ are initialized with 1 and are trained with Stochastic gradient descent(SGD). In addition, to make the pipeline general and less influenced by attacks, the training of reasoning component is independent with that of predictors. We propose the *Pseudo-training* for the reasoning component using the sampled outputs of predictors, rather than their true outputs.

Concretely, given the ground truth label y_{gt}, we denote the grounding predictor vector as $\boldsymbol{s}_g$, where the correct binary values are assigned to each predictor output variable satisfy the true logic rules as shown in Figure 8 in Appendix B.4. We then sample a set of predictor vectors $\{\boldsymbol{s}_t\}$ as training inputs based on $\boldsymbol{s}_g$: if the the original value is 1, we will sample from the Uniform distribution $U(0.5, 1)$; otherwise sample from $U(0, 0.5)$. Such Pseudo-training has several advantages: (1) the training samples will not be affected by adversarial inputs, (2) the sampled training inputs are balanced and thus avoid biases induced by imbalanced training, (3) such Pseudo-training is very efficient especially when the input is high-dimensional data which enables CARD on large-scale datasets.

Owing to the space limit, the detailed Pseudo-codes for the training of predictors and the reasoning part are deferred to Appendix B.4.

3.3 Theoretical Analysis of CARD

We let $\boldsymbol{f} = (f_0, f_1, ..., f_n)^\intercal$, therefore, given $x \in \mathbb{R}^d$, $r(\boldsymbol{f}(x)) \in \mathcal{Y}$ represents the output of the CARD pipeline.

Robustness Certification Protocol. We certify the robustness of CARD following the standard randomized smoothing (Cohen et al., 2019) as follows: with Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ added to clean input x_0, suppose the ground-truth class is $y_0 \in \mathcal{Y}$, we make prediction as $P_A = \mathbb{P}\left(r(\boldsymbol{f}(x+\varepsilon)) = y_0\right)$. Intuitively, P_A is the probability that the CARD returns the correct prediction under input noise. Then, from (Cohen et al., 2019), the Gaussian smoothed classifier $g(x) := \arg\max_{y\in\mathcal{Y}} \mathbb{P}\left(r(\boldsymbol{f}(x+\varepsilon)) = y\right)$ which can be certified: under no further assumption, if the correct prediction probability is P_A, the certified ℓ_2 radius of g at input x is $r = \sigma\Phi^{-1}(P_A)$, i.e, for any x' such that $\|x' - x\|_2 \le r$, $g(x') = g(x)$.

Theoretical Analysis. We theoretically compare CARD with a single model (only using the main predictor) and the commonly-used weighted ensemble (Liu et al., 2020; Yang et al., 2021). We let benign input be $x_0 \in \mathbb{R}^d$, ground-truth label be $y_0 \in \mathcal{Y}_0$, and ε be the added noise sampled from a general noise distribution (not necessarily Gaussian). We make the following assumptions.

Assumption 3.1 (Main Predictor). *The main predictor f_0's correct prediction probability under ϵ is p: $\mathbb{P}_\varepsilon(f_0(x_0+\varepsilon) = y_0) = p$.*

Definition 2 (Weighted Ensemble). The weighted ensemble is composed of $(n+1)$ submodels $\{f^{(i)}\}_{i=1}^{n+1}$, where each submodel $f^{(i)} : \mathbb{R}^d \to \mathcal{Y}_0$ is associated with a confidence margin function $h^{(i)} : \mathbb{R}^d \to \mathcal{Y}_0$. The submodel predicts y_0 if and only if the confidence margin function is positive. The weighted ensemble is defined by static weights $\boldsymbol{w} \in \mathbb{R}_+^{n+1}$: $\mathcal{M} : \mathbb{R}^d \to \mathcal{Y}_0$, where

$$\mathcal{M}(x) = y_0 \iff \sum_{i=1}^{n+1} \boldsymbol{w}_i h^{(i)} > 0. \tag{5}$$

Remark. The margin function can be viewed as the margin of the submodel's logit layer confidence between the true class y_0 and the runner-up class y_1, i.e., $f^{(i)}(x) = F^{(i)}(x)_{y_0} - F^{(i)}(x)_{y_1}$ where $y_1 = \arg\max_{y\neq y_0} F^{(i)}(x)_y$ and $F^{(i)}$ is the logit layer confidence of $f^{(i)}$. Since the weighted ensemble typically sums up the logit layer confidence and predicts the class with the highest summed confidence (Liu et al., 2020), to ensure the correct prediction y_0, the sum of margin needs to be positive, i.e., Equation (5).

Assumption 3.2 (Weighted Ensemble). *Each submodel's correct prediction probability under ε is p:*

$$\forall i \in \{1, 2, \ldots, n+1\}, \mathbb{P}_\varepsilon(f^{(i)}(x_0+\varepsilon) = y_0) = \mathbb{P}_\varepsilon(h^{(i)}(x_0+\varepsilon) > 0) = p. \tag{6}$$

We assume each $h^{(i)}$ follows the same Gaussian distribution:

$$h^{(i)}(x_0+\varepsilon) \sim \mathcal{N}(\mu, \sigma^2). \tag{7}$$

To model the high transferability across submodels (Tramèr et al., 2017), for any $i \neq j$, we assume the Pearson correlation coefficient between $h^{(i)}(x_0+\varepsilon)$ and $h^{(j)}(x_0+\varepsilon)$ is a positive number ρ, and all $\{h^{(i)}(x_0+\varepsilon)\}_{i=1}^{n+1}$ follow multivariate Gaussian distribution:

$$p_{ij} = \rho \in (0, 1). \tag{8}$$

In general, p_{ij} is relatively high as observed in (Papernot et al., 2016).

Assumption 3.3 (CARD). *We assume that under input noise, each of the n knowledge predictors has $q > 1/2$ correct prediction probability on its own true label:*

$$\forall i \in \{1, 2, \ldots, n\}, \mathbb{P}_\varepsilon(f_i(x_0+\varepsilon) = y_i) = q. \tag{9}$$

The CARD predicts correctly if and only if the main predictor is correct or more than half of the knowledge predictors are correct:

$$r(\boldsymbol{f}(x_0+\varepsilon)) = y_0 \iff f_0(x_0+\varepsilon) = y_0 \vee |\{f_i : f_i(x_0+\varepsilon) = y_i, 1 \le i \le n\}| \ge n/2. \tag{10}$$

Since different tasks are given to different predictors, the correlation between different predictors is low. Thus, we assume that the events of correct predictions are mutually independent:

$$\mathrm{p}_i := \mathbb{I}[f_i(x_0+\varepsilon) = y_i], \{\mathrm{p}_i\}_{i=0}^{n} \textit{ are mutually independent}. \tag{11}$$

Remark. Among the above assumptions, for fairness comparison, we let the correct prediction probability of the main predictor to be always p. For weighted ensemble, the correlation between submodel predictions is high. For CARD, the correlation between predictors is generally low (except for certain blocks which are logically correlated). The intuition behind the robustness of CARD is that given strong logic rules encoded by CARD, some incorrect predictors can be efficiently corrected.

Theorem 2. *Under Assumptions 3.1 to 3.3,*

$$\mathbb{P}_\varepsilon(\mathcal{M}(x_0+\varepsilon)=y_0) \le \Phi\left(\sqrt{\rho^{-1}}\Phi^{-1}(p)\right), \tag{12}$$

$$\mathbb{P}_\varepsilon(r(\boldsymbol{f}(x_0+\varepsilon))=y_0) \ge 1-(1-p)\exp\left(-2n\left(q-1/2\right)^2\right), \tag{13}$$

where Φ *is the Gaussian CDF. In particular,*

$$\mathbb{P}_\varepsilon(\mathcal{M}(x_0+\varepsilon)=y_0) > p, \quad \mathbb{P}_\varepsilon(r(\boldsymbol{f}(x_0+\varepsilon))=y_0) > p. \tag{14}$$

We prove the theorem in Appendix A.3. The proof is based on an affine projection of multivariable normal distribution and Hoeffeding's inequality.

Remark. Recall that the main predictor has correct prediction probability p. Thus, we find that:

1. Both weighted ensemble and CARD are more robsut than the single model.
2. Along with the increase of the number of models, the correct prediction probability of weighted ensemble cannot approach 1, while our CARD can approach a correct prediction probability of 1 with a sufficiently large number of predictors, i.e., when n is large, CARD's correct prediction probability is *always* higher than the weighted ensemble. This exactly matches our empirical findings in Section 4.4.

In addition, when $\varepsilon \sim \mathcal{N}(0,\sigma^2)$, as mentioned above, if the correct prediction probability is P_A, the certified ℓ_2 radius of the smoothed single model/ensemble/CARD at input x_0 is $r=\sigma\Phi^{-1}(P_A)$. Since r monotonically increases with P, *the above findings about the correct prediction probability directly transfers to the certified* ℓ_2 *radii*. Note that if Assumptions 3.1 to 3.3 do not hold, since we compute the robustness certification following (Cohen et al., 2019), the certification is still valid.

4 Experimental Evaluation

In this section, we would demonstrate experimental evaluation of CARD on high-resolution dataset Animals with Attributes(AwA2) and Word50. As we can see, with the knowledge integration, CARD achieves *significantly* higher certified robustness than the state-of-the-art baselines. We also provide some interesting understandings and conclusions of different knowledge.

4.1 Experimental Setup

We conduct experiments on AwA2 and Word 50 datasets. We defer more details to appendix B.1. The model architectures chosen for all the predictors in dataset AwA2 is ResNet-50 (He et al., 2016); for dataset Word50, we follow the similar setting in Chen et al. (2015), and choose multi-layer perceptrons (MLPs) with rectified linear units (ReLU) as the predictors. All the experiments are run on four NVIDIA 2080 Ti GPUs.

Evaluation Metric. Following Cohen et al. (2019), we would report the *certified accuracy* for CARD and other pipelines under Gaussian noise with different varianceσ and under different ℓ_2 radius R.

Baselines. We consider four state-of-the-art certifiably robust baselines: (1) *Gaussian smoothing* (Cohen et al., 2019) trains smoothed classifiers by directly adding the Gaussian noise during training; (2) *SmoothAdv* (Salman et al., 2019a) employs adversarial training to improve the certified robustness based on the Gaussian smoothing; (3) *Consistency* (Jeong and Shin, 2020) adds a consistency regularization term to the standard Gaussian smooth training; (4) *SWEEN* (Liu et al., 2020) employs weighted ensemble to improve the certified accuracy.

4.2 Experimental Results on AwA2

Logic Relationships. For the hierarchy logic rules, similar to Deng et al. (2014), we utilize WordNet (Miller, 1995) to build a hierarchy tree by iteratively searching the inherited hypernyms of the 50 leaf classes. We perform the main classification task on the 28 internal nodes to integrate knowledge relationships with both their child and parent nodes. We train 50 knowledge predictors to classify each leaf node as a binary classification, and build the logical clauses like *blue whale* $\Longrightarrow$ *cetacean* and *cetacean* $\Longrightarrow$ *aquatic* on both directions.

For attribute-based logic rules, we train 85 knowledge predictors for classifying each attribute annotated in AwA2 as a binary classification with logic clauses such as *blue whale* $\implies$ *live in ocean*. Training details in appendix B.4.

Certification Results. The certification results of CARD and baselines are shown in Figure 3 (row 1). To demonstrate the flexibility of CARD, we train the main predictor with different baselines denoted as CARD-x, where "x" represents the training algorithm. All the knowledge predictors are still trained with the standard Gaussian Smoothing (Cohen et al., 2019) for simplicity. As we can see, CARD performs quite stable with different training algorithms for the main predictor, and CARD *significantly* improves the certified accuracy under different perturbation radii with different smoothing levels. The number of base models used in SWEEN ensemble is 20 here.

4.3 Experimental Results on word50

Logic Relationships. We perform both word-level and character-level classifications as the main prediction task on Word50. For the word-level classification, we aim to classify the word of the input images, and each image consists of 5 characters. The main predictor is trained to classify the 50 words, and we train 5 knowledge predictors to classify the character for each position. We denote *Pos(i,x)* as the random variable representing the confidence of predicting the character at ith position of the input image as character "*x*". Since we find that in this dataset, the identification of the characters on three positions is enough to determine the whole word, we build logic clauses like *Pos(1,s)* $\wedge$ *Pos(3,a)* $\wedge$ *Pos(4,c)* $\implies$ *"Snack"* as the "hierarchy" knowledge direction, and clauses like *"Snack"* $\implies$ *Pos(2,n)* $\wedge$ *Pos(5,k)* as the "attribute-based" knowledge direction as shown in Appendix B.3. In total, we build the clauses for any three or four characters classified on the five positions for the "hierarchy" direction and it would result in 15 clauses for each word. On the "attribute-based" direction, we build clauses for any one or two characters classified on the five positions and it would result in 15 clauses for each word. Then the total number of clauses is 1600, the detailed construction can be found in Appendix B.3. Finally, for the classification on the *character* level, we would first use the clauses defined before to identify the current input word and thus predict the corresponding character at a specific position. Training details in appendix B.4.

Certification Results. The certified accuracy results on the word-level and character-level classification are shown in the second and third row of Figure 3, respectively. Since the size of the input image is small, the Gaussian noise levels $\sigma \in \{0.12, 0.25, 0.50\}$. Similarly, as we can see, CARD outperforms all other baselines significantly, under different perturbation radii and smoothing noise levels. All the knowledge predictors used in CARD are trained under Gaussian noise for simplicity. The number of base models used in SWEEN ensemble is 6, which equals the number of predictors in CARD.

4.4 Ablation Studies

Utilities of Different Knowledge. Here we aim to explore the utilities of different knowledge in terms of improving the certified robustness of the whole CARD pipeline. On dataset AwA2, we let "attrP" represent the clauses built for the positive attributes, $e.g.$, *persian cat* $\implies$ *white*, and "attrN" for the clauses built for the negative attributes, $e.g.$, *persian cat* $\implies$ $\neg$ *aquatic*. The hierarchy clauses "hierP" and "hierN" are defined similarly. Then, clearly the number of clauses defined by "attrPN" and "hierPN" for each class is 85 and 50 respectively, which represent the clauses for both the positive and the negative clauses. The number of clauses defined by "attrP" and "hierP" for each class depends on the number of its attributes and child nodes.

The comparison of certified robustness enabled by different knowledge is shown in Figure 4. The main predictor of CARD here is trained under Gaussian smoothing. The "main+attrPN+hierPN" represents the whole logical relations we used for CARD in Section 4.2, and more training details could be found in Appendix B.3. As we can see, given the same main predictor, the certified robustness is different by using different types of knowledge. Interestingly, although the number of "hierP" is 50, which is relatively small, such knowledge is quite effective for improving the certified robustness. The similar exploration on Word50 could be found in Appendix B.5.

Number of Knowledge Predictors. As shown in our theoretical analysis (Theorem 2), the certified robustness of CARD would improve with the increase of the number of knowledge predictors. Here we aim to verify it with experiments. First, we calculate the importance scores of clauses trained under "CARD-main+attrPN+hierPN", and for each knowledge predictor, we calculate the averaged importance score based on the clauses related it. Next, we select the most influential top$k$ knowledge

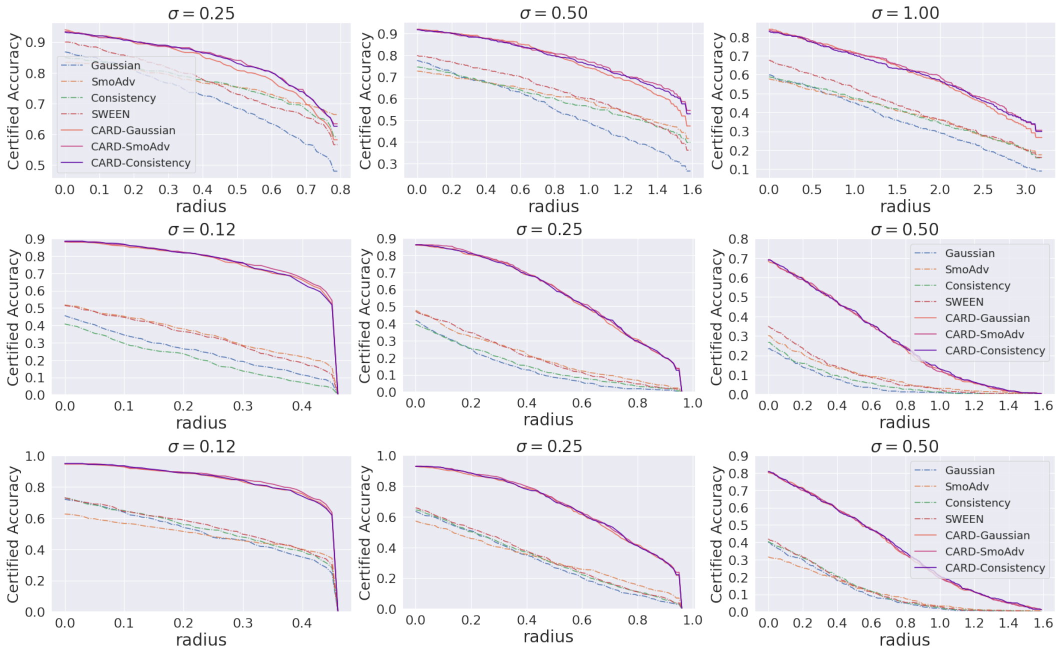

Figure 3: Certified accuracy under different perturbation radii on different datasets: AwA2 (Row1), word-level classification on Word50 (Row2), and character-level classification on Word 50 (Row3).

predictors with the maximal averaged importance scores to build CARD. As shown in Figure 5, the certified robustness of CARD indeed improves with the increase of k. On the contrary, for the weighted ensemble, such as SWEEN, when we increase the number of base models in the ensemble from 3 to 20 the improved certified robustness is marginal as shown in Figure 9 in Appendix B.5, which again verifies our theoretical analysis in Theorem 2.

In addition, we explore the different strategies to relax the reasoning component by defining and training the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ differently. We find that different relaxation strategies would further improve the certified robustness of CARD and we defer the results and discussion to Appendix B.5.

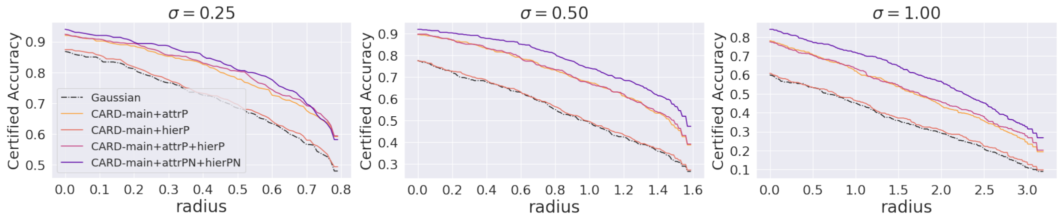

Figure 4: Certified accuracy of CARD using different knowledge on AwA2.

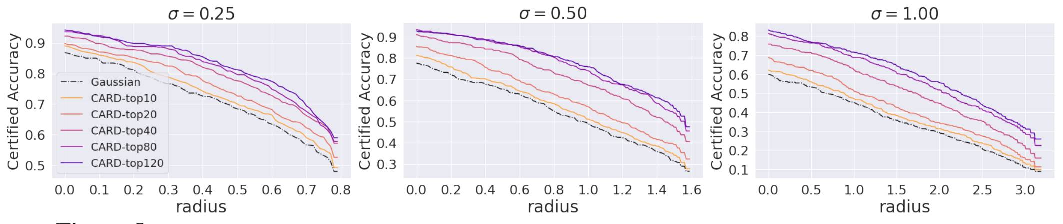

Figure 5: Certified accuracy of CARD using different number of knowledge predictors on AwA2.

5 CONCLUSIONS

In this work, we propose the first knowledge-enabled certifiably robust machine learning pipeline. We show both theoretically and empirically that the proposed pipeline CARD achieves high certified robustness. We expect our framework and findings will inspire interesting future directions on leveraging domain knowledge to make machine learning models more trustworthy.

Ethics Statement. In this paper, we prove the first framework of a certifiably robust machine learning pipeline via knowledge integration. We aim to leverage human knowledge to improve the trustworthiness of machine learning models, and we do not expect any ethics issues raised by our work.

In addition, for many ML algorithms, there could be potential bias issues which usually come from the imbalance of training datasets. In other words, there are limited features or less samples can be used for minority groups. To reduce such bias and encourage fair learning in practice, instead of directly training based on pure data, CARD introduces prior domain knowledge and adopts simulated training so as to sample the balance data for training.

On the other hand, since the proposed framework CARD would leverage existing or crafted domain knowledge or commonsense knowledge rules if such knowledge rules are biased, it is possible to induce biased learning outcomes for the learning pipeline. As a result, we need to follow the standard policies and auditing principles to collect and filter the used knowledge rules for the robust learning systems. Besides, given that the proposed CARD does not require a complete set of knowledge rules since a subset of knowledge rules are sufficient to avoid inconsistency (i.e., potential adversarial behaviors), it is possible to filter the used knowledge rules to ensure that they are not biased and satisfy existing ethical requirements.

Reproducibility Statement. All theorem statements are substantiated with rigorous proofs in our Appendix. We have uploaded the source code as the supplementary material for reproducibility purposes.

References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *Journal of Machine Learning Research*, 18:1–67, 2017.

Islam Beltagy, Katrin Erk, and Raymond Mooney. Probabilistic soft logic for semantic textual similarity. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1210–1219, 2014.

Luca Bertinetto, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A Lord. Making better mistakes: Leveraging class hierarchies with deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12506–12515, 2020.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.

Ashok K Chandra and David Harel. Horn clause queries and generalizations. *The Journal of Logic Programming*, 2(1):1–15, 1985.

Liang-Chieh Chen, Alexander Schwing, Alan Yuille, and Raquel Urtasun. Learning deep structured models. In *International Conference on Machine Learning*, pages 1785–1794. PMLR, 2015.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys (CSUR)*, 33(3):374–425, 2001.

Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *European conference on computer vision*, pages 48–64. Springer, 2014.

Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

Matteo Fischetti and Jason Jo. Deep neural networks as 0-1 mixed integer linear programs: A feasibility study. *arXiv preprint arXiv:1712.06174*, 2017.

James Foulds, Shachi Kumar, and Lise Getoor. Latent topic networks: A versatile probabilistic programming framework for topic models. In *International Conference on Machine Learning*, pages 777–786. PMLR, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H Hovy, and Eric P Xing. Harnessing deep neural networks with logic rules. In *ACL (1)*, 2016.

Jongheon Jeong and Jinwoo Shin. Consistency regularization for certified robustness of smoothed classifiers. In *34th Conference on Neural Information Processing Systems (NeurIPS) 2020*. NeurIPS committee, 2020.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

Chizhou Liu, Yunzhen Feng, Ranran Wang, and Bin Dong. Enhancing certified robustness via smoothed weighted ensembling. *arXiv preprint arXiv:2005.09363*, 2020.

Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint arXiv:1706.07351*, 2017.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

Hadi Salman, Jerry Li, Ilya P Razenshteyn, Pengchuan Zhang, Huan Zhang, Sébastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *NeurIPS*, 2019a.

Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems*, 32:9835–9846, 2019b.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *AAAI*, 2018a.

Chaowei Xiao, Jun Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018*, 2018b.

Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 2740–2748, 2015.

Greg Yang, Tony Duan, J Edward Hu, Hadi Salman, Ilya Razenshteyn, and Jerry Li. Randomized smoothing of all shapes and sizes. In *International Conference on Machine Learning*, pages 10693–10705. PMLR, 2020.

Zhuolin Yang, Linyi Li, Xiaojun Xu, Bhavya Kailkhura, Tao Xie, and Bo Li. On the certified robustness for ensemble models and beyond. *arXiv preprint arXiv:2107.10873*, 2021.

A Detailed Proofs

Usually, the clauses built with attribute-based knowledge is Type 1 or Type 2, and the clauses built with hierarchy knowledge is Type 3 or Type 4. In addition to the knowledge predictor, we still have one main predictor, and the clauses built for it would be the form of $m = t$ where m is the one dimension random variable in the confidence vector output from the main predictor and t is the discrete random variable representing the corresponding class. We will decompose this logical representation into two clauses: $m \implies t$ and $t \implies m$, and force them share the same importance score w. So the main predictor would thus contribute $2|\mathcal{Y}_0|$ clauses totally. This new representation is actually equivalent to the original logical expression $m = t$ for inference. One specific example is provided in Appendix A.2.

A.1 1-NN Reasoning Representation Proof

Proof of Theorem 1. Since the clauses built for the main predictor could be reduced to a special case of the Type 1 and Type 3 clauses, then here we first show that the calculation of the weighted penalty score of these four types clauses shown on Equation (3) could be represented as the form of an importance score times a linear function of the elements of the vector $\boldsymbol{z} = [\boldsymbol{s}; \boldsymbol{y}_j]$ with a ReLU activation. Assume the discrete one dimension random variable t represents a specific target class and $\boldsymbol{y}_j$ is the one-hot label vector. Denote the number of picked subset $s_i, s_j, ..., s_k$ as n and the number of total clauses as L.

Type 1 and 3 clause: $s_i \vee s_j \vee ... \vee s_k = \min\{s_i + s_j + ... + s_k, 1\}$, notice $t \in \{0, 1\}$, so $\neg t \sqcup (s_i \vee s_j \vee ... \vee s_k) = \max\{t - \min\{(s_i + s_j + ... + s_k), 1\}, 0\} = \max\{t - (s_i + s_j + ... + s_k), 0\}$. And it is similar for $\neg(s_i \vee s_j \vee ... \vee s_k) \sqcup t = \max\{\min\{(s_i + s_j + ... + s_k), 1\} - t, 0\} = \max\{(s_i + s_j + ... + s_k) - t, 0\}$.

Type 2 and 4 clause: $\neg t \sqcup (s_1 \wedge s_2 \wedge ... \wedge s_n) = \max\{t - (s_i + s_j + ... + s_k)/n, 0\}$ and $\neg(s_1 \wedge s_2 \wedge ... \wedge s_n) \sqcup t = \max\{(s_i + s_j + ... + s_k)/n - t, 0\}$.

Although we use the operator $\wedge$ as the linear approximation of the conjunction operator $\&$ in Type 2 and 4 clause, the conclusion still holds if we replace the $\wedge$ back to $\&$:

First, it is easy to extend the binary calculation of $s_i \,\&\, s_j = \max\{s_i + s_j - 1, 0\}$ to the multiple one $s_i \,\&\, s_j \,\&\, ... \,\& s_k = \max\{(s_i + s_j + ... + s_k) - (n - 1), 0\}$. Denote the importance score for this clause as w, then the weighted penalty score of $\neg t \sqcup (s_i \,\&\, s_j \,\&\, ... \,\&\, s_k)$ is $w \max\{t - \max\{(s_i + s_j + ... + s_k) - (n - 1), 0\}\}$. Next, we would break down this calculation into two parts with the same importance vector: $w \max\{t - (s_i + s_j + ... + s_k) + (n - 1), 0\}$ and $-w \max\{(n - 1) - (s_i + s_j + ... + s_k), 0\}$. It is quick to verify that the sum of these two parts is equal to the original one. This trick also applies to the case $\neg(s_i \,\&\, s_j \,\&\, ... \,\&\, s_k) \sqcup t$.

Notice the discrete random variable t which represents a target class here is just one dimension of the one-hot vector $\boldsymbol{y}_j$. Then as we could see, the original calculation of the clause score for this class could be represented as $\boldsymbol{w} \max\{\boldsymbol{G}\boldsymbol{z}^T + \boldsymbol{\beta}, \boldsymbol{0}\}$, where $\boldsymbol{w}$ is the importance scores with L dimensions, $\boldsymbol{G}$ is the coefficient matrix with shape $L \times (m + |\mathcal{Y}_0|)$ and the value of its element $g_{i,j}$ is determined by the coefficient of the element z_j appeared in the i_{th} clause, $\boldsymbol{\beta}$ is the corresponding bias vector whose non-zero elements are from the clauses related to the operator $\&$. Notice, if the element z_j is not picked in the i_{th} clause, then g_{ij} is 0. Most of the time, each clause would only build the logical relation among small part of the random variables of the $\boldsymbol{z} = [\boldsymbol{s}; \boldsymbol{y}_j]$, so the $\boldsymbol{G}$ is quite sparse in practice.

So the left problem now is just to remove the latent $\boldsymbol{y_j}$ in $\boldsymbol{z}$, and thus instead of iteratively assigning the $\boldsymbol{y_j}$ from $(1, 0, ..., 0)$ to $(0, 0, ..., 1)$ to get the clause score for each class, we could get a clearer expression for better optimization. The idea is also quite intuitive, just blocking the matrix $\boldsymbol{G}$ first:

$$\boldsymbol{G}\boldsymbol{z} + \boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{E} \end{pmatrix} \begin{pmatrix} \boldsymbol{s}^T \\ \boldsymbol{y}_j^T \end{pmatrix} + \boldsymbol{\beta} = \boldsymbol{C}\boldsymbol{s}^T + \boldsymbol{E}\boldsymbol{y}_j^T + \boldsymbol{\beta}, \tag{15}$$

where the shape of $\boldsymbol{C}$ is $L \times m$ and the shape of $\boldsymbol{E}$ is $L \times |\mathcal{Y}_0|$.

Now we want to remove the $\boldsymbol{y}_j$ here, and we could do all the assignments of it at one time by using matrix multiplication. Denote $\boldsymbol{W}$ as diag$(\overbrace{\boldsymbol{w}, \boldsymbol{w}, \cdots, \boldsymbol{w}}^{|\mathcal{Y}_0| \text{ times}})$ and $\boldsymbol{A}$ as diag$(\overbrace{\boldsymbol{C}, \boldsymbol{C}, \cdots, \boldsymbol{C}}^{|\mathcal{Y}_0| \text{ times}}) \times$

$\overbrace{\begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} \end{pmatrix}}^{|\mathcal{Y}_0| \text{ times}}{}^T$, where $\boldsymbol{I}$ is the identity matrix with shape $m \times m$. Further define the matrix $\boldsymbol{B}$ as a column vector with $|\mathcal{Y}_0|$ dimensions, where $\boldsymbol{B}_{i\times L:(i+1)\times L} = \boldsymbol{\beta} +$ the $(i+1)$ th column of $\boldsymbol{E}$, $\forall i \in \{0, \ldots, |\mathcal{Y}_0| - 1\}$. Then $\boldsymbol{W} \max\{\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}, \boldsymbol{0}\}$ would directly return the column vector with $|\mathcal{Y}_0|$ dimensions and each dimension represents the corresponding clause score for this class. In practice, the multiplication would be implemented parallelly. $\square$

A.2 Example.

Consider the illustration in Figure 2, and denote the sensing vector $\boldsymbol{s}$ as $[m_1, m_2, a_1, a_2, h_1, h_2]$, where each random variable represents the confidence of the corresponding object in Figure. Further, we denote the assignment of the target label *cat* as t_1 and the *Cetacean* as t_2. As mentioned before, the clause $m_i = t_i$ built for the main predictor would be decomposed to two clauses $m_i \implies t_i$ and $t_i \implies m_i$ with the same importance score. Then we introduce eight simple clauses for example as follows:

$$\begin{array}{l} w_1: \text{ (1) } t_1 \implies m_1 : \neg t_1 \sqcup m_1 = \max(t_1 - m_1, 0) \\ w_1: \text{ (2) } m_1 \implies t_1 : \neg m_1 \sqcup t_1 = \max(m_1 - t_1, 0) \\ w_2: \text{ (3) } t_2 \implies m_2 : \neg t_2 \sqcup m_2 = \max(t_2 - m_2, 0) \\ w_2: \text{ (4) } m_2 \implies t_2 : \neg m_2 \sqcup t_2 = \max(m_2 - t_2, 0) \\ w_3: \text{ (5) } t_1 \implies a_1 : \neg t_1 \sqcup a_1 = \max(t_1 - a_1, 0) \\ w_4: \text{ (6) } t_2 \implies a_2 : \neg t_2 \sqcup a_2 = \max(t_2 - a_2, 0) \\ w_5: \text{ (7) } h_1 \implies t_1 : \neg h_1 \sqcup t_1 = \max(h_1 - t_1, 0) \\ w_6: \text{ (8) } h_2 \implies t_2 : \neg h_2 \sqcup t_2 = \max(h_2 - t_2, 0) \end{array} \tag{16}$$

Denote

$$\boldsymbol{w} = (w_1, w_1, w_2, w_2, w_3, w_4, w_5, w_6), \boldsymbol{G} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}. \tag{17}$$

Now define $\boldsymbol{z} := [\boldsymbol{s}; t_1; t_2]$, then $\boldsymbol{w} \cdot \max(\boldsymbol{G}\boldsymbol{z}^T, \boldsymbol{0})$ just represents the clause score when given the target one-hot label vector $[t_1; t_2]$. Further denote $\boldsymbol{I}$ as the identity function with shape 6×6, $\boldsymbol{A}$ as $\text{diag}(G[:6], G[:6]) \times [\boldsymbol{I}, \boldsymbol{I}]^T$ where $G[:i]$ means the first i columns of $\boldsymbol{G}$, $\boldsymbol{W}$ as $\text{diag}(\boldsymbol{w}, \boldsymbol{w})$. Since there is no bias constant in this simple example, i.e., the bias vector $\boldsymbol{\beta}$ brought from Equation (16) is a null column vector. Then the matrix $\boldsymbol{B}$ here is simply the concatenation of the last two columns vectors of $\boldsymbol{G}$ with shape 16×1. The final corresponding reasoning model is just the matrix multiplication form:

$$r(\boldsymbol{s}) = \arg\min\{\boldsymbol{W} \max(\boldsymbol{A}\boldsymbol{s}^T + \boldsymbol{B}, \boldsymbol{0})\} \tag{18}$$

A.3 Proof in Theoretical Analysis of CARD

Proof of Theorem 2. We prove these equations sequentially.

(I)

First, combining Equation (6) and Equation (7) we have

$$\Phi\left(-\frac{\mu}{\sigma}\right) = 1 - p \Longrightarrow p = 1 - \Phi\left(-\frac{\mu}{\sigma}\right) = \Phi\left(\frac{\mu}{\sigma}\right).$$

To simplify the notation, we use random variable $\mathbf{h}_i$ to represent $h^{(i)}(x_0 + \varepsilon)$ when ε is sampled from the noise distribution, and $\mathbf{h} \in \mathbb{R}^{n+1}$ is the random vector that concatenates each $\mathbf{h}_i (1 \le i \le n+1)$ together. According to Assumption 3.2,

$$\mathbf{h} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\mu} = (\mu, \mu, \cdots)^\intercal \in \mathbb{R}^{n+1}$ and $\boldsymbol{\Sigma} = \sigma^2 \rho \mathbf{1}\mathbf{1}^\intercal + \sigma^2 (1 - \rho) \boldsymbol{I}_{n+1}$, where $\mathbf{1} = (1, 1, \cdots)^\intercal \in \mathbb{R}^{n+1}$ and $\boldsymbol{I}_{n+1}$ is an $(n+1) \times (n+1)$ identity matrix. Now we can infer the distribution of random variable

$$\mathrm{s} := \sum_{i=1}^{n} \boldsymbol{w}_i \mathbf{h}_i = \boldsymbol{w}^\intercal \mathbf{h}.$$

According to the affine transformation rule of multivariate Gaussian distribution, s follows Gaussian distribution $\mathcal{N}(\mu_s, \sigma_s^2)$ where

$$\mu_s = \boldsymbol{w}^\intercal \boldsymbol{\mu} = \mu \|\boldsymbol{w}\|_1,$$

$$\begin{aligned}\sigma_s^2 = \boldsymbol{w}^\intercal \boldsymbol{\Sigma} \boldsymbol{w} &= \sum_{i=1}^{n+1}\sum_{j=1}^{n+1} w_i w_j \boldsymbol{\Sigma}_{ij} = \sum_{i=1}^{n+1}\sum_{j=1}^{n+1} w_i w_j \sigma^2 \rho + \sum_{i=1}^{n+1} w_i^2 \sigma^2 (1-\rho) \\ &= \sigma^2 \rho \|\boldsymbol{w}\|_1^2 + \sigma^2(1-\rho)\|\boldsymbol{w}\|_2^2.\end{aligned}$$

Therefore,

$$\begin{aligned}\mathbb{P}_\varepsilon(\mathrm{s} \geq 0) &= 1 - \Phi\left(-\frac{\mu_s}{\sigma_s}\right) = \Phi\left(\frac{\mu_s}{\sigma_s}\right) \\ &= \Phi\left(\frac{\mu \|\boldsymbol{w}\|_1}{\sqrt{\sigma^2\rho\|\boldsymbol{w}\|_1^2 + \sigma^2(1-\rho)\|\boldsymbol{w}\|_2^2}}\right) = \Phi\left(\frac{\sigma\Phi^{-1}(p)\|\boldsymbol{w}\|_1}{\sqrt{\sigma^2\rho\|\boldsymbol{w}\|_1^2 + \sigma^2(1-\rho)\|\boldsymbol{w}\|_2^2}}\right) \\ &= \Phi\left(\frac{\Phi^{-1}(p)}{\sqrt{\rho + (1-\rho)\|\boldsymbol{w}\|_2^2/\|\boldsymbol{w}\|_1^2}}\right).\end{aligned} \tag{19}$$

By Definition 2, the correction prediction probability of weighted ensemble is

$$\mathbb{P}_\varepsilon(\mathcal{M}(x_0+\varepsilon) = y_0) = \mathbb{P}_\varepsilon\left(\sum_{i=1}^{n+1} \boldsymbol{w}_i h^{(i)}(x_0+\varepsilon) > 0\right) = \mathbb{P}_\varepsilon(\mathrm{s} > 0).$$

With Equation (19), we get

$$\mathbb{P}_\varepsilon(\mathcal{M}(x_0+\varepsilon) = y_0) = \Phi\left(\frac{\Phi^{-1}(p)}{\sqrt{\rho + (1-\rho)\|\boldsymbol{w}\|_2^2/\|\boldsymbol{w}\|_1^2}}\right) \leq \Phi\left(\sqrt{\frac{1}{\rho}}\Phi^{-1}(p)\right).$$

This is Equation (12) in theorem statement.

(II)

In CARD, recall that random variable $\mathrm{p}_i = \mathbb{I}[f_i(x_0+\varepsilon) = y_i], 0 \leq i \leq n$ (Equation (11)). According to Assumption 3.3,

$$\begin{aligned}\mathbb{P}_\varepsilon(r(\boldsymbol{f}(x_0+\varepsilon)) = y_0) &= 1 - \mathbb{P}_\varepsilon\left(\mathrm{p}_0 = 0 \vee \sum_{i=1}^{n} \mathrm{p}_i < n/2\right) \\ &= 1 - (1-p)\cdot\mathbb{P}\left(\sum_{i=1}^{n}\mathrm{p}_i < n/2\right) && \text{(by mutual independence)} \\ &\overset{(*)}{\geq} 1 - (1-p)\cdot\exp\left(-2n\left(q - \frac{1}{2}\right)^2\right), && \text{(by Hoeffding's inequality)}\end{aligned} \tag{20}$$

which is Equation (13) in theorem statement. The $(*)$ can use the Hoeffding's inequality since $\{\mathrm{p}_i\}_{i=1}^n$ are 1) mutually independent; 2) bounded by $[0,1]$; and 3) have expectation q. It is possible to further tighten the inequality using the tail bound of binomial distribution.

(III)

Now we prove Equation (14). Recall that in Equation (19),

$$\mathbb{P}_\varepsilon(\mathrm{s} \geq 0) = \Phi\left(\frac{\Phi^{-1}(p)}{\sqrt{\rho + (1-\rho)\|\boldsymbol{w}\|_2^2/\|\boldsymbol{w}\|_1^2}}\right).$$

Since $\boldsymbol{w}_i > 0$ by Definition 2, we know $\|\boldsymbol{w}\|_2 < \|\boldsymbol{w}\|_1$ and hence

$$\sqrt{\rho + (1-\rho)\|\boldsymbol{w}\|_2^2/\|\boldsymbol{w}\|_1^2} < 1.$$

Thus,

$$\mathbb{P}_\varepsilon(\mathcal{M}(x_0+\varepsilon) = y_0) = \mathbb{P}_\varepsilon(\mathrm{s} \geq 0) = \Phi\left(\frac{\Phi^{-1}(p)}{\sqrt{\rho + (1-\rho)\|\boldsymbol{w}\|_2^2/\|\boldsymbol{w}\|_1^2}}\right) > \Phi\left(\Phi^{-1}(p)\right) = p. \quad (21)$$

Meanwhile, for CARD, recall Equation (20):

$$\mathbb{P}_\varepsilon(r(\boldsymbol{f}(x_0+\varepsilon)) = y_0) = 1 - (1-p)\cdot\mathbb{P}\left(\sum_{i=1}^{n} \mathrm{p}_i < n/2\right).$$

Since $\mathbb{E}\sum_{i=1}^n \mathrm{p}_i = nq > n/2$ by Assumption 3.3,

$$sP\left(\sum_{i=1}^{n} \mathrm{p}_i < n/2\right) < 1.$$

Therefore,

$$\mathbb{P}_\varepsilon(r(\boldsymbol{f}(x_0+\varepsilon)) = y_0) > 1 - (1-p) = p. \quad (22)$$

The Equations (21) and (22) are combined to Equation (14) in the theorem statement. □

B Experiment Details

B.1 Datasets

To integrate different knowledge as first-order logic rules to demonstrate the effectiveness of CARD, we first conduct experiments with the dataset Animals with Attributes (AwA2) Xian et al. (2018), which consists of 37322 (resized to 224 × 224) for classification. The whole dataset contains 50 animal classes and provides 85 binary class attributes for each class, e.g., for *persian cat*, "furry" is *yes* and "stripes" is *no*. In this dataset, some popular classes like *horse* has 1645 examples but some less popular classes like *mole* only has 100 samples. Such data imbalanced phenomenon is common in practice and it is quite interesting to see if the prior domain knowledge can help to handle it, given that the number of samples with specific attributes is still sound for us to train a good attribute knowledge predictor. As a result, it is easy to see the bottleneck of purely training from data or simply stacking plenty of main models to do ensemble owing to the inability of handling such imbalanced data. Even at the theoretical level, as shown in Theorem 2, along with the increase of the number of models, the correct prediction probability of weighted ensemble cannot approach 1, while our CARD can approach a correct prediction probability of 1. Besides, with more models, the performance of the ensemble will converge as theoretically proven by Yang et al. (2021).

In addition, we also conduct experiments on Word50 dataset (Chen et al., 2015), which is created by randomly selecting 50 words and each consisting of five characters. All the character images are of size 28 × 28 and perturbed by scaling, rotation, and translation. The background of the characters is blurry by inserting different patches, which makes it a quite challenging task. Sometimes it is even hard for human to recognize the characters. The interesting property of this dataset is that the character combination is given (50 known words) as the prior knowledge, which can be integrated into our CARD. The training, validation, and test sets contain $10,000$, $2,000$ and $2,000$ variations of words styles, respectively. In our experiment, we would certify the robustness on both *word*-classification and *character*-classification levels.

B.2 Training and Certification Details.

Training and Certification Details on AwA2 For the training of the knowledge predictors, in every training epoch, we would sample half images with the attribute/hierarchy from the training data and sample half images without it. Therefore, we do not need to use all the training data for the knowledge predictors, which would save a lot of time compared to the training of the main predictor and encourage the generalization. The predictor vector $\boldsymbol{s}$ here could be represented as $[\boldsymbol{m};\boldsymbol{a};\boldsymbol{h}]$,

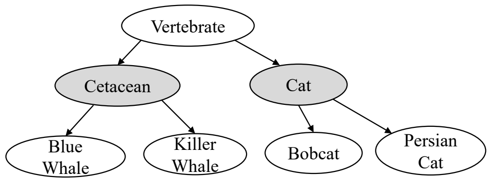

Figure 6: Hierarchy tree of AwA2, and the main classification task is on the gray node level.

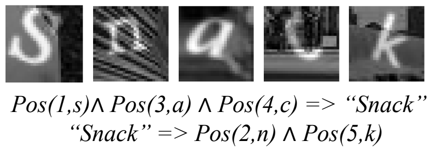
Pos(1,s)∧ Pos(3,a) ∧ Pos(4,c) => "Snack"
"Snack" => Pos(2,n) ∧ Pos(5,k)

Figure 7: Knowledge and logic rules on Word50.

where $\boldsymbol{m}$ is the output of the main predictor with 28 dimensions, $\boldsymbol{a}$ is output of the attribute predictors with 85 dimensions, and $\boldsymbol{h}$ is the output of the hierarchy predictors with 50 dimensions. Notice, for each image sampled from the leaf node, it is relabeled to its parent node, and the grounding value of $\boldsymbol{a}$ still depends on its original annotation of the attributes for the training of the importance score $\boldsymbol{w}$. During training, we randomly sample $10,000$ simulated data for each class, and the number of the training epoch is set to 10, the batch size is set to 2048, then the whole training for $\boldsymbol{w}$ could be finished within 5 minutes. Following the certification algorithm in Cohen et al. (2019) and for saving the certification time on the total 135 knowledge predictors, all the results were certified with the $N = 10,000$ samples and failure probability $\alpha = 0.001$. For the baseline SWEEN, we train 20 main models with standard Gaussian Smoothing to do the weighted ensemble.

Training and Certification Details on Word50. We randomly select 10 images for each word from the test dataset for certification , and the total number of certified images here is 500. The main predictor is trained to classify the input word consisted of five characters, and all the knowledge predictors here are trained to classify the 26 characters. The predictor vector $\boldsymbol{s}$ could be represented as $[\boldsymbol{m}; \boldsymbol{e}_1; ...; \boldsymbol{e}_5]$, where $\boldsymbol{m}$ is the output of the main predictor with 50 dimensions, $\boldsymbol{e}_i$ is the output of the knowledge predictors which is responsible for the classification of the character at the ith position. The hyperparameters for training the importance score $\boldsymbol{w}$ are the same in AwA2. All the results were certified with $N = 100,000$ samples of smoothing noise.

B.3 Built Clauses Details.

To construct different logic rules for the target prediction, we focus on the classification on the internal nodes as shown in Figure 6. The sampled images and example logic clauses on dataset Word50 can be found at Figure 7. Besides, as mentioned in the Appendix A, each clause for $m = t$ would be converted into two clauses in our allowed clauses type. So the number of the clauses built in the "CARD-main+attrPN+hierPN" is $28 \times 2 + 28 \times (85 + 50) = 3836$, then similarly the number of clauses built on Word50 would be $50 \times 2 + 50 \times (15 + 15) = 1600$. These two are just the clauses used in the CARD in Figure 3. For other knowledge, the number of clauses for each class is dependent on the number of its own positive attributes and the child nodes it has. The number of the clauses defined by the knowledge "CARD-main+attrP" is 1074, and for "CARD-main+hierP", it is 106. Then for "CARD-main+attrP+hierP", the total number is 1124.

B.4 Training and Certification Details.

The training of the predictors and the reasoning part is independent, as mentioned in Section 3.2. For the former, the corresponding training process is shown Algorithm 1, and notice that we expect the attribute predictors to own high TP rates, and the hierarchy predictors own high TN rates. For the training of the reasoning part, we adopt pseudo-training for reducing the imbalance and bias in the actual training dataset as shown in Algorithm 3 and the illustration of the process of generating simulated training data for the reasoning part is shown in Figure 8. The embedding process of the knowledge rules to the 1-NN neural network r is detailed discussed in Appendix A.1, and an intuitive example is provided at Appendix A.2. The final prediction process of our whole pipeline during the testing stage is shown in Algorithm 4.

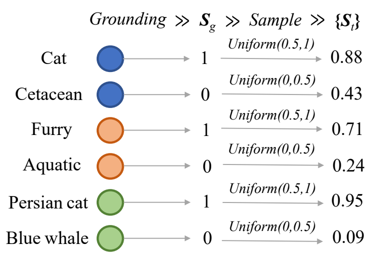
Grounding ≫ S_g ≫ Sample ≫ $\{S_t\}$
Cat 1 Uniform(0.5,1) 0.88
Cetacean 0 Uniform(0,0.5) 0.43
Furry 1 Uniform(0.5,1) 0.71
Aquatic 0 Uniform(0,0.5) 0.24
Persian cat 1 Uniform(0.5,1) 0.95
Blue whale 0 Uniform(0,0.5) 0.09

Figure 8: The process of sampling the Pseudo-training dataset $\{\boldsymbol{s}_t\}$, given the truth label *cat*.

For AwA2, we randomly sample 80% images from each leaf node as the training data, and pick 10 images for each leaf node from the remaining unsampled images for certification. Following the standard setting Cohen et al. (2019) we certify 500 images with noise sampling size $\alpha = 0.001$.

Algorithm 1 The training process of knowledge predictors for one batch.

```
Input: Base model $f$, images $x_1, x_2, ..., x_n$, binary ground labels $y_1, y_2, ..., y_n$, noise level $\sigma$.
 1: Drawn $n$ samples of noise, $\epsilon_1, ..., \epsilon_n \sim \mathcal{N}(0, \sigma^2 I)$.
 2: Obtain The predictions $f(x + \epsilon_1), ..., f(x + \epsilon_n)$.
 3: if the predictor is trained for classifying attribute then
 4:     for $i = 1$ to $n$ do
 5:         if $y_i == 1$ then
 6:             Loss weight $w_i$ of $BCE(f(x + \epsilon_i), 1) \leftarrow 2$
 7:         else if $y_i == 0$ then
 8:             Loss weight $w_i$ of $BCE(f(x + \epsilon_i), 0) \leftarrow 1$
 9:         end if
10:     end for
11: else if the predictor is trained for classifying hierarchy then
12:     for $i = 1$ to $n$ do
13:         if $y_i == 1$ then
14:             Loss weight $w_i$ of $BCE(f(x + \epsilon_i), 1) \leftarrow 1$
15:         else if $y_i == 0$ then
16:             Loss weight $w_i$ of $BCE(f(x + \epsilon_i), 0) \leftarrow 2$
17:         end if
18:     end for
19: end if
20: Final weighted $BCE$ loss $l = \sum_{i=1}^{n} \frac{w_i}{\sum_{i=1}^{n} w_i} .BCE(f(x + \epsilon_i), y_i)$.
21: Computes the gradient w.r.t. $l$ and update the parameters of $f$.
```

Algorithm 2 Get the grounding predictor vector s_g.

```
Input: Ground label $y_0 \in \mathcal{Y}$., $m$ attributes, $k$ hierarchies.
Output: The corresponding grounding predictor vector $s_g$.
 1: Initialize a zero vector $s_g$ with $|\mathcal{Y}|$ dimensions.
 2: $s_g[y_0] \leftarrow 1$.
 3: for $i = 1$ to $m$ do
 4:     if the class of label $y_0$ has $i_{th}$ attribute. then
 5:         $s_g \leftarrow [s_g, 1]$
 6:     else
 7:         $s_g \leftarrow [s_g, 0]$
 8:     end if
 9: end for
10: for $k = 1$ to $k$ do
11:     if the class of label $y_0$ has $j_{th}$ hierarchy. then
12:         $s_g \leftarrow [s_g, 1]$
13:     else
14:         $s_g \leftarrow [s_g, 0]$
15:     end if
16: end for
17: return $s_g$
```

Algorithm 3 The training process of the reasoning part for one batch.

```
Input: 1-NN reasoning neural network $r$, uniformly sampled class labels $y_1, y_2, ..., y_n \in \mathcal{Y}$, the
    corresponding grounding predictor vector $s_{g_1}, s_{g_2}, ..., s_{g_n}$ got from Algorithm 2.
 1: Initialize predictor zero vectors $s_{t_i}, i \in \{1, 2, ..., n\}$ with the same dimension of $s_{g_i}$.
 2: for $i = 1$ to $n$ do
 3:     $s_{t_i}[s_{g_i} == 0] = Uniform(0, 0.5)$
 4:     $s_{t_i}[s_{g_i} == 1] = Uniform(0.5, 1)$
 5: end for
 6: Obtain the penalty scores $r(s_{t_1}), ..., r(s_{t_n})$.
 7: Final loss $l = \sum_{i=1}^{n} CrossEntropy(-r(s_{t_i}), y_i)/n$.
 8: Computes the gradient w.r.t. $l$ and update the parameters of $r$.
```

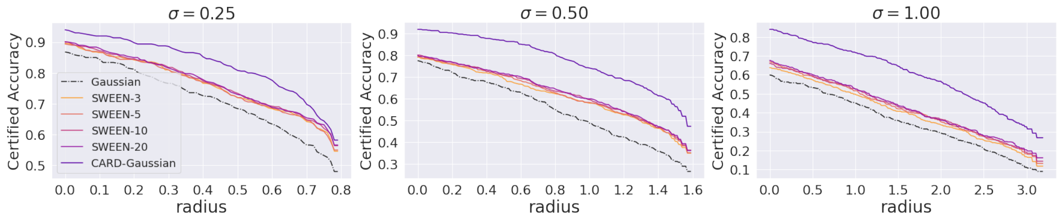

Figure 9: Comparison of certified accuracy with SWEEN containing different number of base models on AwA2.

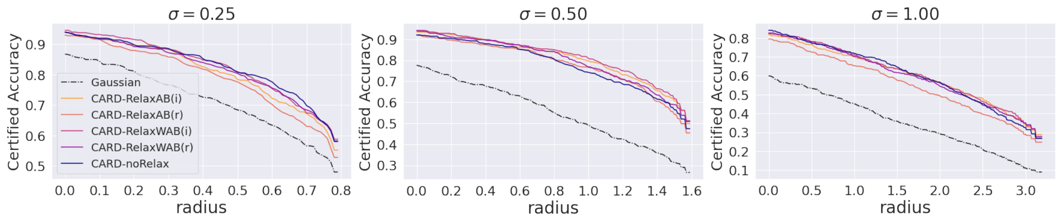

Figure 10: Certified accuracy of CARD under different relaxation of the matrices on AwA2.

During training, we randomly sample $10,000$ Pseudo-training data for each class, and the number of the training epoch is 10 with batch size. 2048. The whole Pseudo-training process for training the clause importance weights $\boldsymbol{w}$ can be finished within 5 minutes.

For Word50, We randomly select 10 images for each word from the test dataset for certification , and the total number of certified images here is 500. The main predictor is trained to classify the input word consisted of five characters, and all the knowledge predictors here are trained to classify the 26 characters. The predictor vector $\boldsymbol{s}$ could be represented as $[\boldsymbol{m}; \boldsymbol{e}_1; ...; \boldsymbol{e}_5]$, where $\boldsymbol{m}$ is the output of the main predictor with 50 dimensions, $\boldsymbol{e}_i$ is the output of the knowledge predictors which is responsible for the classification of the character at the ith position. The hyperparameters for training the importance score $\boldsymbol{w}$ are the same in AwA2. All the results were certified with $N = 100,000$ samples of smoothing noise.

Algorithm 4 The prediction process of the whole pipeline during testing stage.

```
Input: Image $x$, predictors $f_0, f_1, ..., f_n$, 1-NN reasoning neural network $r$.
Output: Predicted label.
  Initialize the predictor vector $s$ with $f_0(x)$.
  for $i = 1$ to $n$ do
    $s \leftarrow [s, f_i(x)]$
  end for
  Obtain the penalty scores $r(s)$.
  return $\arg\min r(s)$
```

B.5 Extra Ablation Study.

Number of Knowledge Predictors. The ensemble is a method of gathering a bunch of similar models and doing a simple weighted sum without inherent logical relations; however, although we also have a bunch of models, there is some prior domain knowledge. Each model is responsible for different tasks, and the first-order logical relations between the outputs of different models are built for more robust prediction. Then one interesting question is *With the increase of the number of the predictors, what is the trend for the certified accuracy?* When we gradually increase the number of base models used in weighted ensemble from 3 to 20, the improved performance shown in Figure 9 is quite marginal, which means the performance of this ensemble way has converged and this phenomenon has also been theoretically proven by Yang et al. (2021). However, with the domain knowledge and logic, such phenomenon is alleviated as shown in Figure 5.

Different knowledge used on Word50. The input image size to the main predictor is $5 \times 28 \times 28 = 3920$, and the image input to the knowledge predictor is a single character image, so the corresponding input size is $28 \times 28 = 784$. All the predictor models here are trained under standard Gaussian

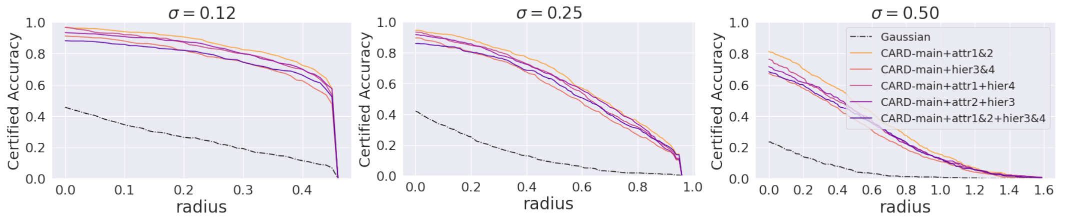

Figure 11: Certified accuracy of CARD using different knowledge on Word50 for the word-classification task.

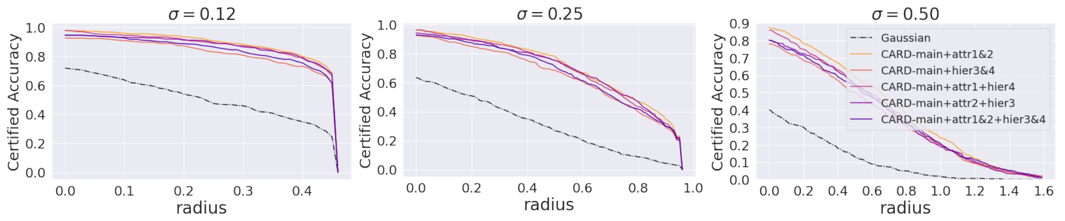

Figure 12: Certified accuracy of CARD using different knowledge on Word50 for the character-classification task.

Smoothing for simplicity. We use "attr$k$" to denote the clause whose number of the elements in the tail is k. For example, "attr2" just means the clauses built like *"Snack"* $\implies$ *Pos(1,s)* $\wedge$ *Pos(3,a)*. Similarly, we use "hier$k$" to denote the clause whose number of the elements in the head is k. So "hier4" is used to represent the clauses like *Pos(1,s)* $\wedge$ *Pos(2,n)* $\wedge$ *Pos(3,a)* $\wedge$ *Pos(4,c)* $\implies$ *"Snack"*. The "attr1&2" is used to represent the clauses contains both the "attr1" and "attr2", the definition of "hier3&4" could be similarly obtained. Then as we can see, even given the same input predictor vector, with the different knowledge and the variation of the clauses we use, the final certified accuracy still could be strongly influenced as shown in Figure 11 and Figure 12, which is strong and compelling evidence for showing the potential of CARD.

Relaxation of Reasoning Component. Another interesting exploration is about the relaxation of the matrices shown in the reasoning model r, i.e., the matrices $\boldsymbol{W}$, $\boldsymbol{A}$, and $\boldsymbol{B}$. During our formal experiment setting, the matrix $\boldsymbol{A}, \boldsymbol{B}$ are constrained and determined by the pre-defined logic relations. So they are fixed and not trained, however, it is reasonable if we also train them. Take a simple example, if we denote the confidence variables for *persian cat*, *white* and *furry* as p, w and f respectively. Then, the penalty score for the clause *persian cat* $\implies$ *white* $\wedge$ *furry* is $\max(p - w/2 - f/2, 0)$. However, if *furry* is a more important attribute to distinguish the persian cat from other animals, then it is reasonable to change the coefficient of w and f to $-1/3$ and $-2/3$ respectively, which means the attribute *furry* would influence the penalty score more. In addition, the matrix $\boldsymbol{W}$ could also be relaxed and the relaxed matrices are all trained by SGD, the corresponding results are shown in Figure 4. The "*i*" in the parentheses means the matrices are initialized by the given logical relations("main+attrPN+hierePN"), the "*r*" means the matrices are initialized randomly. It shows that sometimes such relaxations may help, but they could not be controlled well owing to the lack of explicit knowledge and logic encoding. The further design of the optimization of these matrices is still an open problem and would be explored in the future.