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 $\text{diag}(\overbrace{\boldsymbol{w}, \boldsymbol{w}, \cdots, \boldsymbol{w}}^{|\mathcal{Y}_0| \text{ times}})$ and $\boldsymbol{A}$ as $\text{diag}(\overbrace{\boldsymbol{C}, \boldsymbol{C}, \cdots, \boldsymbol{C}}^{|\mathcal{Y}_0| \text{ times}}) \times$

$\overbrace{(\ \boldsymbol{I}\quad \boldsymbol{I}\quad \cdots\quad \boldsymbol{I}\ )}^{|\mathcal{Y}_0| \text{ times}}{}^{T}$, 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. □

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:

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

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

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

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,

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

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.

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 $[\boldsymbol{m}; \boldsymbol{a}; \boldsymbol{h}]$, 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

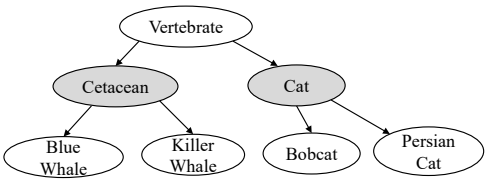

Figure 6: Hierarchy tree of AwA2, and the main classification task is on the gray node level.

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.

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.

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 $\alpha = 0.001$. 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. More training details are deferred to Appendix B.2.

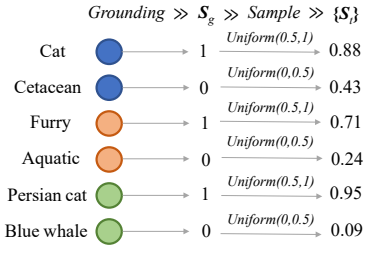

Figure 8: The process of sampling the Pseudo-training dataset $\{\boldsymbol{s}_t\}$, given the truth label *cat*.

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.

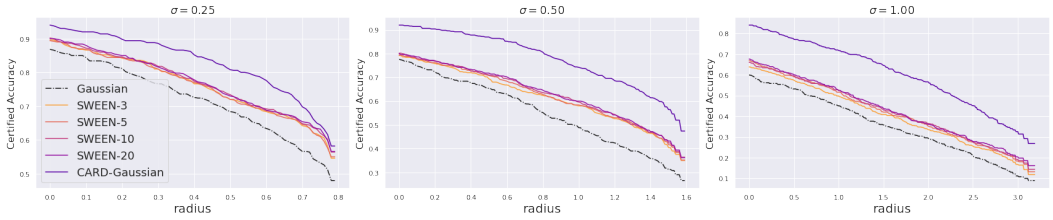

Figure 9: Comparison of certified accuracy with SWEEN containing different number of base models on AwA2.

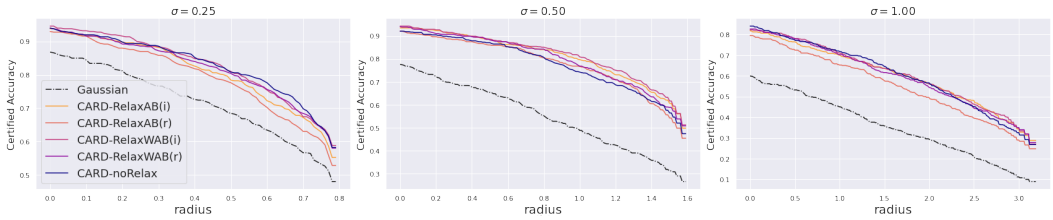

Figure 10: Certified accuracy of CARD under different relaxation of the matrices on AwA2.

B.5 Extra Ablation Study.

Number of Knowledge Predictors. 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 Smoothing for simplicity. We use "attr1" to mean the clauses built like *"Snack"* $\Longrightarrow$ *Pos(1,s)* and "attr2" to mean the clauses built like *"Snack"* $\Longrightarrow$ *Pos(1,s)* $\wedge$ *Pos(3,a)*, and similarly, the "hier3" is used to represent the clauses like *Pos(1,s)* $\wedge$ *Pos(3,a)* $\wedge$ *Pos(4,c)* $\Longrightarrow$ *"Snack"* and "hier4" is used to represent the clauses like *Pos(1,s)* $\wedge$ *Pos(2,n)* $\wedge$ *Pos(3,a)* $\wedge$ *Pos(4,c)* $\Longrightarrow$ *"Snack"*. The "attr12" is used to represent the clauses contains both the "attr1" and "attr2", the definition of "hier34" 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* $\Longrightarrow$ *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.

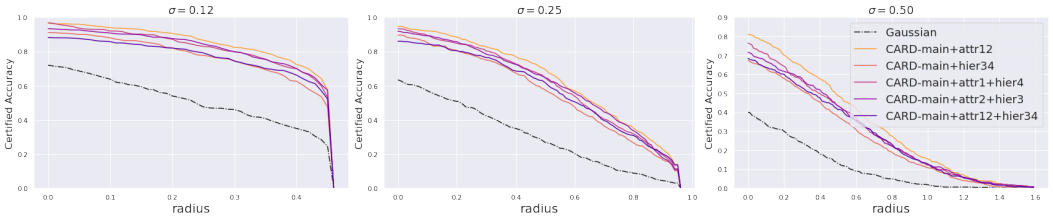

Figure 11: Certified accuracy of CARD using different knowledge on Word50 for the word-classification task.

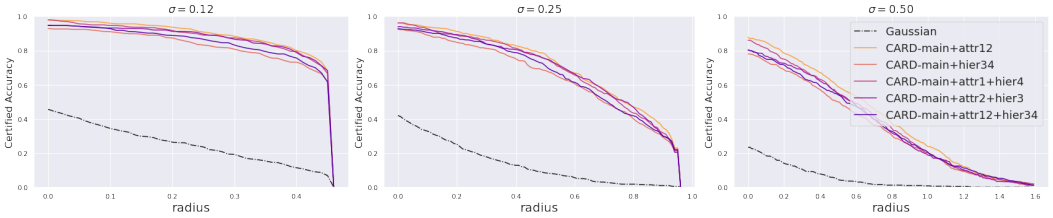

Figure 12: Certified accuracy of CARD using different knowledge on Word50 for the character-classification task.